University of Alberta

Mechanical Design Using the Genetic Algorithm

by

Kent West ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Mechanical Engineering

Edmonton, Alberta
Spring 2001

0-612-60513-2

Kent West

Kent West
6743-87 Street
Edmonton, Alberta
Canada, T6E 2Y7

Date: April 2, 2001

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Mechanical Design Using the Genetic Algorithm** submitted by Kent West in partial fulfillment of the requirements for the degree of **Master of Science**.

[illegible]
A. William Lipsett

[illegible]
Roger Toogood

[illegible]
J. Fraser Forbes

Date: APRIL 6, 2001

To the pursuit of dreams and enjoyment of the day

Abstract

Mechanism design is concerned with finding combinations of mechanical elements that will perform a specified task. Optimization seeks the solutions to a problem which realize a maximum pay-off. Applying optimization to mechanical design finds the best solutions for design tasks.

Genetic Algorithms (GAs) are a broad class of search techniques based on the process of natural selection. A simple GA is used to find optimal solutions for mechanism design tasks.

An introduction to the GA is followed by performance testing of the algorithm. General guidelines for its use are explored and a method for assessing algorithm performance is tested. The GA is applied to two mechanism design tasks where it is shown that the definition of the objective function and the choice of input parameters is very important.

The ability of the GA to find global optimal solutions to problems with complex solution spaces is demonstrated. Methods to gauge the success of the method and recommendations for further applications are presented.

Acknowledgements

To all those whose patience has been tested time and again during the creation of this.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Document contents

Mechanism design and optimization are two classical scientific problem areas that have intrigued scientists and researchers for ages. Optimization has been a concern from the times of ancient Roman engineers building complex aquaduct systems to modern-day engineers studying things such as vehicle ride safety. Whether the goal is to achieve a consistently accurate duct grade for smooth water transport or come up with a new tire pattern and suspension geometry to maximize vehicle stability, optimization is a powerful tool. It can yield a better design than what already exists while providing insight to how complex problems are most efficiently addressed.

Like optimization, mechanism design is, and has been, prevalent in the sciences for ages. Coming up with new combinations of bodies, constraints and forcing methods has been the catalyst for a large amount of scientific development. Combining optimization and mechanism design is a natural task, almost certainly considered when Archimedes was building the first known water screws until the present day. Whether the optimization method is trial and error or computer-aided logic, the task is the same: to realize the set of parameters that describe the mechanical system which best performs the specified task.

The intent of this study is to determine the applicability of using a simple Genetic Algorithm for mechanical design optimization tasks. A simple Genetic Algorithm is a variant of a class of search and optimization methods collectively known as Genetic search. All Genetic search techniques are based on modeling the process of *natural selection* in a population of individuals. Each individual is a potential solution to the problem; as the population matures and becomes stronger, better solutions to the problem are discovered. An introduction to the Genetic Algorithm (GA) is presented in *Chapter 2, The Genetic Algorithm.* After defining the basic nomenclature used, this chapter steps through a simple example of a GA.

To understand the importance of the settings, or parameters, that the GA uses, *Chapter 3, Performance Testing* attempts to find sets of input parameters that result in efficient algorithm performance. Sets of these parameter values that result in consistently successful searches will be used as guidelines for further experiments. A detailed look at the works of two other researchers who have used the GA for similar problems is then considered, focusing on the sets of parameters that they reported as being optimal ones.

Chapter 4, Four bar linkage path generation synthesis using the GA details using the GA to find the dimensions of four bar linkages for path generation tasks. The four bar linkage geometry needed to guide a point through different sets of specified locations in a plane with a minimum of tracking error is sought. The effect of the GA input parameters, the objective function specification and the design task complexity on algorithm performance is considered.

Understanding some of the traits of the algorithm, such as the input parameters required and ways to formulate meaningful objective functions, the algorithm can then be applied to a 'real' design task. *Chapter 5, Mechanical prosthesis design using the GA* details using the GA to design a simple prosthetic knee joint mechanism. A set of parameter values is found which yields knee movements similar to that of human knee movements. The model complexity is then heightened and a new set of parameters yielding more life-like movement in the model is found.

Conclusions and Recommendations for using the GA as a design tool for mechanical design problems are presented in *Chapter 6*. The strengths and weaknesses of the technique are examined and recommendations for using it and future work are presented.

Chapter 2

The Genetic Algorithm

2.1 Chapter Glossary

- GA - Genetic Algorithm
- SGA - Simple Genetic Algorithm
- F - Individual fitness
- g - Function value at some point $g(x)$
- n - Population size
- $P_{crossover}$ - Crossover probability
- $P_{mutation}$ - Mutation probability
- l - Binary string length
- l_n - Length of binary sub-string n
- δ - Variable discretization value

2.2 Introduction

The Genetic Algorithm (GA) is a simplistic numerical simulation of a natural population evolving over time. In the context of an optimization problem, each individual of the population represents a solution to the problem. The population is subject to operators that are similar to natural evolutionary processes. In particular, each solution has a fitness value, the solutions are combined together in a reproductive-like process and stronger solutions contribute more offspring. GAs come in many shapes and forms; the most basic, the Simple Genetic Algorithm (SGA), will be considered here.

The SGA (Goldberg, 1989a) uses a binary string representation of an individual, a fixed population size, fixed point parameter mapping and simple crossover and mutation. The SGA is a powerful optimization tool, able to find high quality solutions in vast, nonlinear solution spaces. In order to describe it, several important terms must first be defined. The following sections will define what an individual is, what the population is and what are the operators employed to transform one population into the next.

2.2.1 The Individual

An individual of the GA is a possible solution to the problem considered. Each individual, therefore, must contain information that describes the unique properties of the individual. For example, if a GA were used to search for a beer having the lightest color and the smallest alcohol percentage in a collection of beers of the world, then an individual may have two important traits that need to be represented: its color and its alcohol percentage. What follows is a discussion of this.

The Genetic Algorithm works with a population of individuals, each is coded to represent its distinguishing characteristics. The SGA uses a binary string coding such as 1001011010 to describe the individual, where, for example, the first 5 bits describe one distinctive trait and the remaining 5 describe another distinctive trait.

Therefore the individual is completely described by a binary sequence such as 1001011010 that can be decoded to yield its identifying characteristics, or parameters. If individuals are described by many characteristics, then the binary substrings representing each characteristic, or trait, are combined to form the overall identifying binary string of the individual.

To illustrate this, an example using beers of the world will be considered. If a population represents beers of the world (perhaps the beer with the lightest color and lowest alcohol content is being searched for), then each individual contains a description of a particular beer. If we identify beers based solely on their color and alcohol percentage, then the binary string identifying an individual would be composed of the binary string representing the color followed by the binary string representing the alcohol percentage. Each substring within the total string represents a parameter value such as 7%, or 'light brown'. In order to translate between the binary representations and the actual parameter values, a few things must be known, or specified, about each parameter.

Each substring must have fixed upper and lower limits for the parameter value. Knowing these limits and the number of bits that represent each parameter, it is easy to translate from binary string representation to the real value. Hence if the color descriptor of the beer individual is represented using a 5 bit string describing parameter values from light golden to dark brown, then the color of any beer could be one of $2^5 = 32$ shades of brown, where the substring 00000 is light golden in color and the substring 11111 is dark brown. Figure 2.1 shows a schematic of this color scale representation.

Individuals are formed by simply concatenating the descriptors' substrings. Therefore with the beers of the world example, if the second descriptor represents alcohol percentage using 5 bits with limits of 0% to 8%, then we could break down two sample beers of the world as shown in Table 2.1. The Asahi, being light in color and high in alcohol percentage, has an alcohol substring with many high order bits set and a color substring

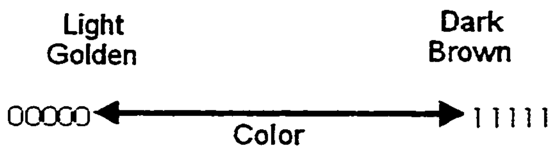

Figure 2.1: The color scale for 'beers of the world' with substring representation using 5 bits.

Name (binary representation)	Color (substring)	Alcohol% (substring)
Asahi Super Dry (0001011101)	Pale golden (00010)	7% (11101)
Newcastle Brown (1110000110)	Chestnut brown (11100)	4% (00110)

Table 2.1: The binary representation of two 'beers of the world'.

with few high order bits set. The binary string representing the Newcastle Brown beer is quite different, being a dark, low-alcohol percentage beer; the substring representing color has many high order bits set while the substring representing alcohol does not.

In this manner, the identifying characteristics of each individual are discretized to fit our classification scheme. The unique binary string depicting our solution is one of two pieces of information that the GA knows about each individual. The other is the fitness of the individual, or how well the solution performs on a specified test.

The notion of fitness is central to the workings of the GA. Each individual has an associated fitness level that indicates how good of a solution it is to the problem at hand. Individuals having different binary string representations perform differently on the fitness tests. Those individuals with desirable traits score well on the tests and are assigned high fitness values. Those not having desirable traits perform poorly on the fitness tests and are assigned low fitness scores.

In the context of our beers of the world example, the fitness of a beer could be based simply on a taster's preference. If, for example, if the judging panel was populated with young Canadian beer drinkers accustomed to light, crisp beers with medium to high alcohol content, the Japanese beer, Asahi Super Dry, would likely score higher than the British beer, Newcastle Brown. The higher score for the Asahi beer would be due to its high alcohol content and light color, which is similar to a typical Canadian beer. The British beer, having a low alcohol content and dark color would likely score lower due to its unfamiliar characteristics.

In summary, the GA knows two things about each individual. The first is its genetic makeup; with the SGA this is a string of binary digits. The second is the fitness associated with the individual's genetic makeup. These are the only two pieces of information that the GA knows and requires of an individual. The collection of individuals, the population, is considered next.

2.2.2 The Population

The population has several important properties that are considered throughout the operation of the GA. The population is simply the collection of n individuals at any point in the run of the GA. The first population is generated randomly, then, as new generations are formed from old generations, the makeup of the population changes. In particular, the individuals in a population in one generation (parents) are used to produce the individuals (children) in the next generation in a manner that generally increases the overall fitness of the new population.

The population is the object that the GA deals with on a broad scale. The GA produces new, stronger solutions to a problem by selecting the better solutions in one population to produce many offspring for the next population. The maximum fitness individual in the population can be found, as can the minimum and average fitness individuals. The individuals selected to produce subsequent generations mix genetic material (ie: bits of their bit strings), yielding children having the good and bad qualities of the par-

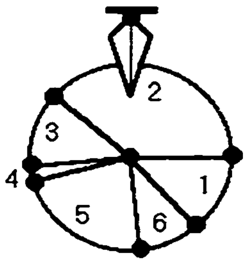

Figure 2.2: Sample roulette wheel for a population of six individuals. Individual 2 has very high fitness with respect to the others, individual 4 has very poor fitness.

ents. As this process occurs, a population of individuals with optimal fitness emerges having near-homogeneous genetic makeup. When this occurs, the average fitness of the population is very similar to the highest fitness in the population - this signals that the population has converged.

2.2.3 The Operators

The GA uses two main operators as it produces new generations: selection and reproduction. Selection dictates which individuals are chosen to produce offspring, reproduction determines how children are formed from the individuals selected as parents.

Selection determines which individuals of a given population will be chosen to produce offspring. This is done probabilistically based on the individual's fitness level. Individuals with high fitness are given a greater probability of being selected to produce offspring. Conversely those with low fitness are given smaller probabilities of being selected to produce offspring. Conceptually this can done by setting up a roulette wheel, where the size of each individual's slot is determined by its fitness. A population of n individuals results in n slots on the wheel and the wheel is spun n times to select n 'parents'.

Figure 2.2 illustrates this procedure for a population size of six individuals

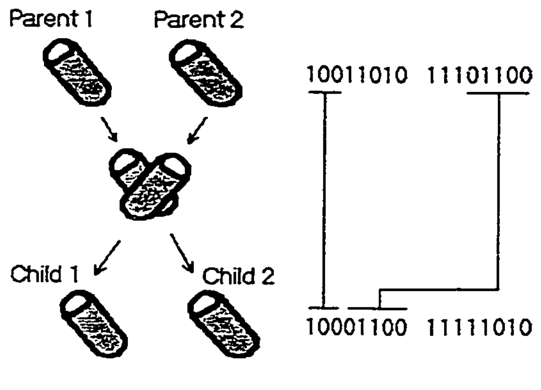

Figure 2.3: Crossing parents to produce children. This recombines the genetic material of the parents in a new manner.

$(n = 6)$. The individuals, labeled 1 to 6, have fitness levels corresponding to the proportion of the roulette wheel that they claim. Hence individual 2 has very high fitness and individual 4 has very low fitness. In the general case, individuals with average fitness would be allotted $1/n$ of the wheel's circumference, hence if the wheel is spun n times, they are expected to be selected, on average, once.

Having selected the members of the previous population that will be parents, offspring are produced by randomly pairing members of the selected group. Children are formed as either perfect copies of, or combinations of, the parent's genetic material. To do this each set of parents has their genetic material crossed over with some probability. Crossover splits the parent strings at the same random location into a head and a tail, as shown in Figure 2.3. The first child is composed of the head of parent 1 and the tail of parent 2. Its sibling is created from the tail of parent 1 and the head of parent 2. This combining of genetic material produces new solutions to the problem having the good and bad traits of each parent. If crossover is not performed then the children are simply exact copies of the parents.

The final means for introducing new genetic material into future populations is by mutation. As each bit is copied from parent to child, there is a small probability that that bit will mutate. With the binary representation of an individual in the SGA, mutation involves simply flipping the bit from either one to zero or vice versa. This serves to introduce new genetic material in a completely random manner, thereby exploring new regions of the solution space that may not be found using crossover alone.

2.3 A simple worked example

To illustrate the detailed workings of the SGA, a simple one variable optimization problem will be considered. The parameter x will be sought that results in the largest value $g(x)$ for the function

$$g(x) = (1 - x) * |\sin(5.1\pi x)| \tag{2.1}$$

in the search interval $0 \leq x \leq 1$. This function, shown graphically in Figure 2.4, was chosen due to its many local maxima and abrupt changes within the solution space.

The abrupt shape of this test function was chosen purposely as traditional gradient based optimization methods would have difficulty finding the global optimum of this function if not given an appropriate initial guess for x. The GA's robust search method will be illustrated by examining its performance for a number of generations. Performance here means equating fitness with the function value as we are searching for the value of x which gives the highest function value. The global optimum for this function in the specified range is located at $x = 0.094$, having a value of $g_{max} = 0.904$. The average value of the function in the solution space is $g_{avg} = 0.318$.

This section will consider first how to set up the GA in terms of input parameters and the variable discretization. How to generate an initial population will then be considered, followed by the generation of subsequent populations. An analysis of the population average and maximum fitness will then be done for increasing generations, highlighting convergence of the algorithm.

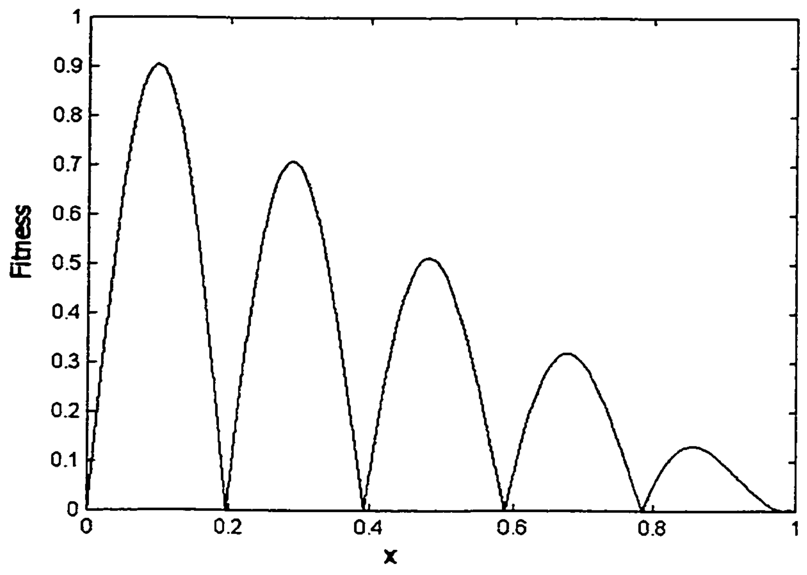

Figure 2.4: The rectified decaying sine wave test function.

2.3.1 Setting up the problem: specifics

The GA requires specification of several input parameters; all affect the searching performance of the GA. The values used for this simple problem are summarized in Table 2.2. With the exception of population size, these values are typical of those found in the literature (see Goldberg (1989a) or Mitchell (1998) for more details).

The population size of $n = 10$ was chosen to be deliberately small for

String Length (l)	6
Population Size (n)	10
$P_{crossover}$	65%
$P_{mutation}$	1.0%
Maximum Generations	40
Convergence Criteria	$F_{avg} \geq 0.98F_{max}$

Table 2.2: GA input parameters.

demonstration purposes. The crossover probability of 65% is typical and means that approximately two-thirds of all children are produced by combining the genetic material of parents and the other one-third are exact copies of their parents. The mutation probability of 1% is also typical, meaning that approximately every one hundredth bit transferred from a parent to a child is mutated. The convergence criteria means that new generations are formed until the population average fitness, F_{avg}, is more than 98% of the maximum fitness, F_{max}, ever found in a population. In other words, new generations are formed until the GA has pushed all of the individuals into one or more very good regions of the solution space. A limit of 40 is set on the number of generations allowed for convergence to occur. Runs of the GA that haven't met the convergence criteria within 40 generations are considered non-converged and are terminated.

The individual string length is set at $l = 6$, hence the variable x being searched is discretized into $2^6 = 64$ levels, the string 000000 representing $x = 0$ and the string 111111 representing $x = 1$. This separates the discrete values of x by steps of size δ, where

$$\delta = \frac{1-0}{64-1} = 0.0159.$$

2.3.2 The initial population

Having specified all of the input parameters required by the GA, an initial population can now be generated. This first population is generated by assigning random values to each bit position of each individual. The process of generating random bit values can be thought of as tossing a fair coin, heads meaning 1, tails meaning 0. In this manner the initial population should contain a rich selection of the solution space, providing an unbiased grounds from which to begin searching.

Producing an initial population (called generation 0) for the example problem having 10 individuals yields the population listed in Table 2.3 and shown graphically in Figure 2.5. The table shows the bit string for each indi-

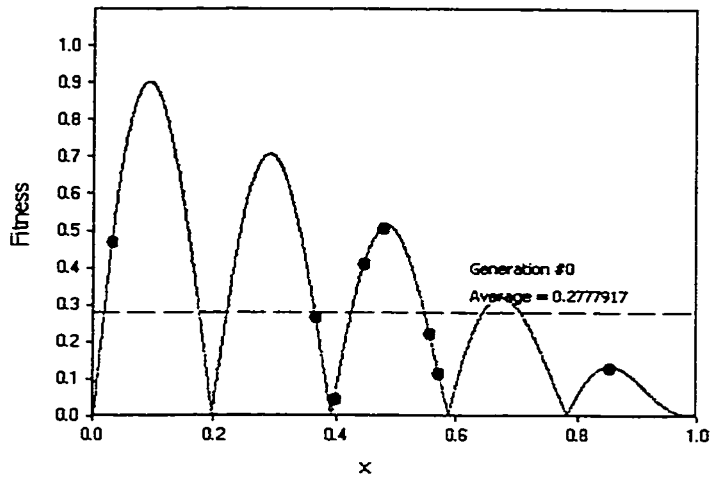

Figure 2.5: The initial population (generation 0).

Individual	Bit String	Parameter Value	Fitness F_i	F_i/F_{avg}	Number of Copies
1	010000	0.0317	0.4713	1.7	2
2	111010	0.3651	0.2686	0.97	1
3	011011	0.8571	0.1310	0.47	1
4	001110	0.4444	0.4115	1.48	1
5	001001	0.5714	0.1159	0.42	0
6	011011	0.8571	0.1310	0.47	0
7	100110	0.3968	0.0431	0.16	0
8	011110	0.4762	0.5102	1.84	3
9	110001	0.5556	0.2240	0.81	0
10	010000	0.0317	0.4713	1.7	2
			$F_{avg} = 0.2778$		

Table 2.3: The initial population (generation 0).

Position	Amount ($\delta = 0.0159$)	Bit	Value
0	$2^0 * \delta$	0	0
1	$2^1 * \delta$	1	0.0317
2	$2^2 * \delta$	1	0.0635
3	$2^3 * \delta$	0	0
4	$2^4 * \delta$	1	0.2540
5	$2^5 * \delta$	1	0.5079
			Sum $= x = 0.8571$

Table 2.4: Decoding the third individual of the initial population, 011011, to find its decoded parameter value.

vidual of the initial population, along with its decoded parameter (x) value, its fitness ($F(x)$) value and its relative ($F(x)/F_{avg}$) fitness and the number of copies that this individual contributes of itself to the next population. Figure 2.5 shows each individual as a solid dot on top of the fitness function, $g(x)$.

In order to assess the fitness of an individual, the decoded parameter value corresponding to each binary string must be calculated. This process is shown in Table 2.4 for individual 3 of the initial population, having a string of 011011. The contribution of each bit position is summed to arrive at a final parameter value. The amount that each bit position[1] contributes to the sum is based on its place, where the first bit position represents the value $2^0 \times \delta$, the second $2^1 \times \delta$ and the n^th $2^{n-1} \times \delta$. Hence the string for individual 3 is decoded to the parameter value $(2^1+2^2+2^4+2^5) \times \delta = 54 \times \delta = 0.8571$. The fitness of the individual is then determined using the fitness function of Equation 2.1, hence $x = 0.8571$ results in a fitness value of $F(0.8571) = F_3 = 0.1310$. A similar procedure is used to assign fitness values to all remaining members of the population.

[1]Note that the bit order convention used here differs from the norm, the lowest order bit being the leftmost bit in the string. This unusual convention is used simply because this was the scheme used in the computer coding of the GA.

Inspecting generation 0 shows a random sampling of the middle of the search region and two pairs of identical individuals (1 and 10, 3 and 6) located at the extents of the domain. The population fitness values range from a low of $F_{min} = 0.0431$ to a high of $F_{max} = 0.5102$, the average fitness value being $F_{avg} = 0.2778$. Comparing the population average fitness to the analytical function average value, $g_{avg} = 0.3179$, indicates that this population is, on the whole, weak. If a sufficiently large group of individuals was selected randomly over the search domain, it would be expected that their average fitness would be very near g_{avg}. This below-average population is therefore a good point from which to begin in order to illustrate the searching power of the GA.

2.3.3 Evolving a new population

The first evolved population (generation 1) must now be produced from the initial population (generation 0). The sequence of steps needed to create a new population is outlined as follows:

1. Generate a roulette wheel based on the relative fitness values of the individuals in the previous population.

2. For a population of size n, spin the roulette wheel n times to select n individuals for the mating pool.

3. Randomly pair individuals of the mating pool, ensuring that members of the pool only mate once.

4. Produce offspring through crossover and mutation.

5. Evaluate the fitness of each member in the new population.

This process will now be considered for the example problem.

Roulette wheel selection

A roulette wheel must be assembled based on the fitness levels of each individual in the population. Knowing the fitness of any individual, F_i, and the sum of the population fitness scores, $\sum_{i=1}^{n} F_i$, the amount of the circumference, C_i, of the roulette wheel allotted to one individual is

$$C_i = \frac{F_i}{\sum_{i=1}^{n} F_i} \times C, \tag{2.2}$$

where $C = 2\pi R$ is the circumference of the circle. In this manner individuals with average fitness are allotted $\frac{C}{n}$ of the roulette wheel.

Setting the wheel up in this manner, the probability of a random spin selecting a given individual is directly proportional to the fitness of the individual. Those individuals with higher fitness values encompass a greater portion of the wheel and hence have a greater probability of contributing more offspring to the next population, while those with lower fitness values probabilistically contribute fewer offspring. The relative fitness value of each individual, F_i/F_{avg}, is therefore an indicator of the number of copies that the individual is expected to contribute to the following population. Those individuals having average fitness, or a relative fitness of 1, are expected to contribute exactly one copy of themselves to the next generation.

Selection into the mating pool

Having setup the roulette wheel, selection can now be performed by spinning the wheel n times to generate a mating pool. This mating pool contains all the parent strings that are used to generate the next population through crossover and mutation. The last column in Table 2.3 shows the number of times that each individual of the initial population was selected into the mating pool needed to produce generation 1. Individuals selected more than once appear as exact copies in the mating pool.

Comparing the number of times that each individual of generation 0 was selected to its relative fitness value illustrates the biased random nature of the

#	Bit String	Value	Fitness F_i	F_i/F_{avg}	Copies	Parents	Cross Point	Mutation
1	011000	0.0952	0.9038	1.85	2	3,10	3	No
2	010011	0.7937	0.0294	0.06	0	3,10	3	No
3	011010	0.3492	0.4148	0.85	0	1,2	2	No
4	110000	0.0476	0.6579	1.35	1	1,2	2	No
5	010000	0.0317	0.4713	0.96	2	1,8	5	No
6	011110	0.4762	0.5102	1.04	1	1,8	5	No
7	010000	0.0317	0.4713	0.96	1	10,8	0	No
8	011110	0.4762	0.5102	1.04	1	10,8	0	No
9	001110	0.4444	0.4115	0.84	1	4,8	2	No
10	011110	0.4762	0.5102	1.04	1	4,8	2	No
				$F_{avg} =$ 0.4891				

Table 2.5: The first evolved population (generation 1).

GA. Individual 7, having a relative fitness value of 0.17, was not selected at all, while individual 1, with a 1.70 relative fitness, has contributed 2 offspring to the next generation. On the whole, individuals with large relative fitness values contribute many copies to the mating pool while those with small values contribute few if any.

Pairing individuals and mating

The members of the mating pool are now paired randomly and mated, involving crossover and mutation. If crossover occurs during mating, the two parent strings are split at the same random location into head and tail sections and the two parts swapped between individuals to create two children having a mixture of the genetic makeup of the parents. If crossover does not occur, the children are exact copies of the two parents. As each bit is copied from parent to child, there is a small probability of mutation occurring. Mutation is performed by simply flipping a one to a zero or vice versa.

This pairing of mates, crossover and mutation results in generation 1,

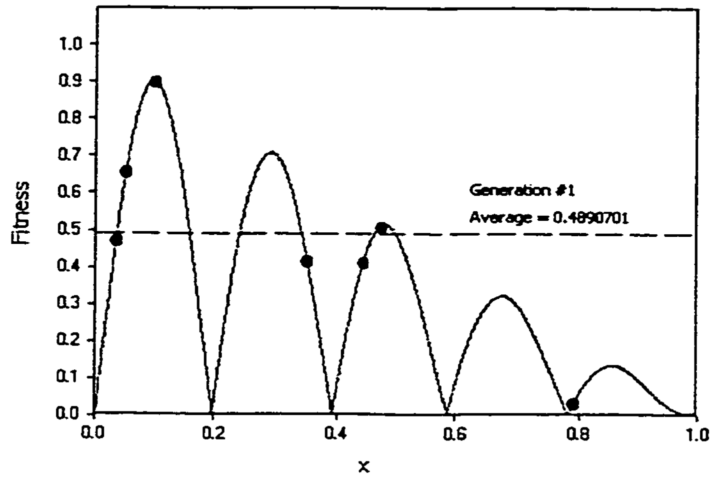

Figure 2.6: The first generation.

shown in Table 2.5 and Figure 2.6. This table includes all the data of Table 2.3 and includes the indices of the parents (from generation 0, shown in Table 2.3) that produced each individual. As well, the bit position at which crossover occurs (0 indicates no crossover occurring) and whether or not mutation occurs is shown.

Considering two individuals of this new population, individuals 1 and 2, it can be seen that members 3 and 10 of generation 0 were the parents, they were crossed over at bit location 3 and no mutation occurred while producing the children. Thus the two solutions to the problem known as individuals 3 and 10 of generation 0, both being rather poor solutions ($F_3 = 0.1310$ and $F_{10} = 0.4713$), were combined to produce a near-optimal and a very poor solution in the following population ($F_1 = 0.9038$ and $F_2 = 0.0294$). Considering the 'number of copies' column in Table 2.5, individual 1 of generation 1 then contributes 2 offspring to the mating pool for the next population while individual 2 doesn't contribute any copies to the next. This represents a somewhat typical example of the GA combining genetic material in new ways to concurrently find better solutions to the problem and eliminate poor

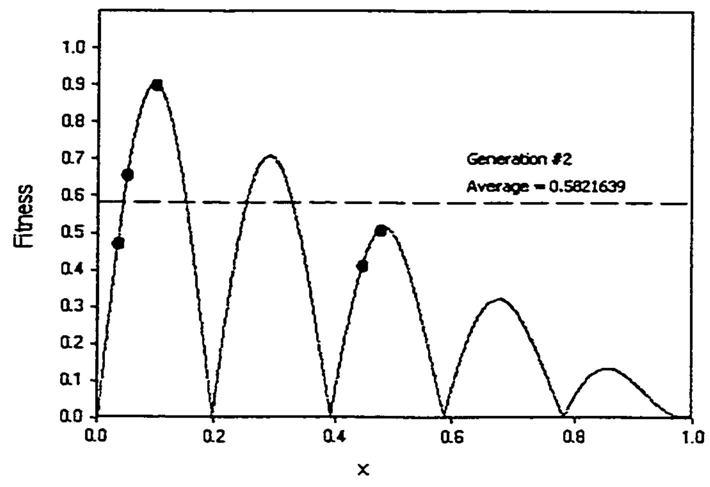

Figure 2.7: The second generation.

ones.

Analyzing generation 1 shows a marked improvement in the population fitness. The population average fitness has increased from 0.2778 to 0.4891. The fitness of the strongest individual has significantly increased from F_8 = 0.5102 in the initial population to F_1 = 0.9038 in the first generation. Scanning the bit string column of Table 2.5 shows that three individuals are identical, having about average fitness. The generation process has therefore increased the overall population fitness and found a very strong solution to the problem. This strong individual subsequently contributes two copies (see the 'Copies' column of Table 2.5) to the mating pool for the next generation. Repeating the generation process has again shown the manner in which the GA combines solutions to explore regions of higher fitness and avoids areas of low fitness.

2.3.4 More generations - continuing on

Additional generations are produced in the same fashion until the convergence criteria is fulfilled or the maximum number of generations is reached.

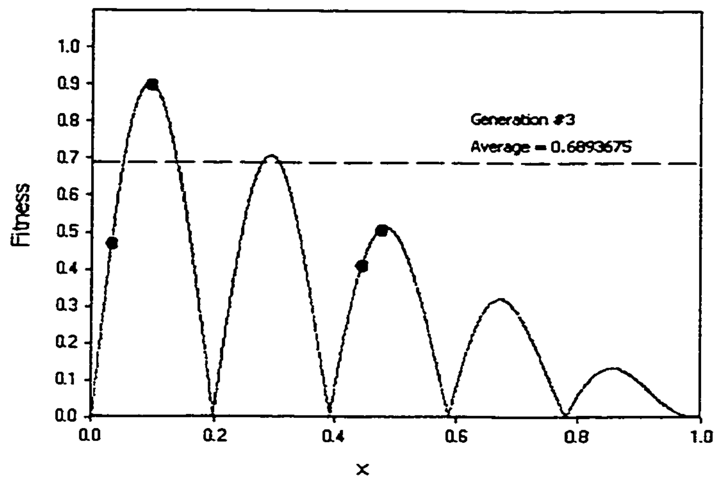

Figure 2.8: The third generation.

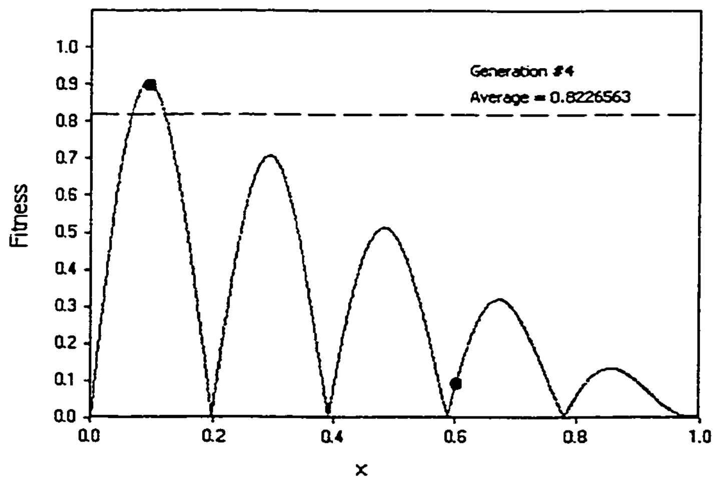

Figure 2.9: The fourth generation.

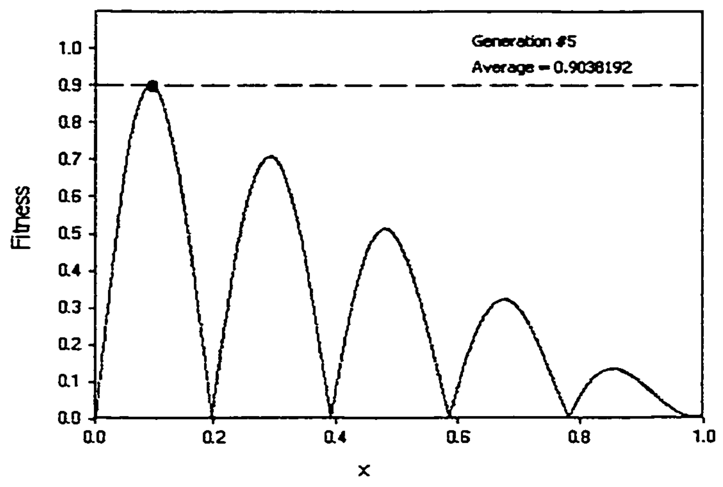

Figure 2.10: The fifth (and final) generation.

Figures 2.7, 2.8 and 2.9 present generations 2, 3 and 4, respectively, showing how the population average fitness steadily increases with further generations. These plots also show how the number of distinct individuals in the population decreases. Figure 2.7 shows only five plotted solution locations, meaning that the other five individuals of generation 2 are exact copies of one of the plotted locations. The following generation, generation 3, has only four distinct locations while generation 4 has just two. The increasing homogeneity of the population's genetic makeup signals convergence of the algorithm.

The final, converged, population, generation 5, is shown in Figure 2.10. All of the individuals have the same genetic makeup, describing one location in the search region having the best fitness ever found in the search. The convergence criteria, $F_{avg} \geq 0.98 * F_{max}$, is therefore satisfied and the algorithm is stopped. The individual having $x = 0.0952$, yielding a fitness value of $F_{x=0.0952} = 0.9038$, is therefore the best solution to the problem. Note that this is not the same as the analytical function maximum value of $g_{max} = 0.9041$ when $x = 0.0940$, as the search variable discretization does not allow for representing this x-value exactly.

2.3.5 Example comments

A peculiar characteristic of the GA search process should be evident from this example. The best solution to the problem was found as early as generation 1, yet the algorithm continued to search for better solutions to the problem, not terminating until all (or, in a more practical problem *nearly* all) of the individuals had this same genetic makeup. At the time of finding the best solution to the problem, the population average fitness of generation 1 was only $F_{avg} = 0.4891$; the convergence criteria had not been fulfilled so the algorithm continued searching for new solutions. This seeming redundancy highlights one of the unique searching abilities of the GA; as there is no way to determine that this is in fact the best solution in the space, the GA continues to search until the convergence criteria is met. It is this process that may find many additional near-optimal solutions in the space, yielding a best solution and several other very good alternatives.

Convergence occurred in this simple example problem, however, this is not always the case. With an optimal set of input parameters (population size, crossover and mutation probabilities) and a reasonably posed problem, convergence on the optimal solution can be expected with some regularity. It is also possible for the GA to converge on a sub-optimal solution (one of the lower peaks in Figure 2.4) in the solution space if the region around the optimal solution is never explored by the algorithm. Alternatively, the algorithm may not converge, stopping only when the maximum number of generations has been reached. It is due to these scenarios that the GA is seldom run just once and the best solution of that run taken as the global optimum solution. The GA should be run several times until completion, where completion is either convergence or reaching the maximum number of generations. Consistent convergence to one or more regions in the search space determines the optimal solution and possibly several very good solutions.

Chapter 3

GA performance testing and applications

3.1 Chapter Glossary

- l - Binary substring length.
- L - Total binary string length.
- n - Population size.
- P_{cross} - Crossover probability.
- $P_{mutation}$ - Mutation probability.

3.2 Introduction and Background

The basics of the GA, particularly the nomenclature and the general concepts, have been considered. One area left unaddressed in the previous chapter is the effect of the input parameters (population size, crossover and mutation probabilities, overall string length, etc.) on the performance of the algorithm. How certain can one be that a large enough population is being used, or the crossover probability is high enough? This chapter will attempt to address the relationship between the input parameters, particularly the population size, crossover probability and mutation probability, on the algorithm's ability to find good solutions to the problems at hand.

Using the simple example problem of the previous chapter, the effect of altering each of the population size, crossover probability and mutation probability will be related to the algorithm's search performance. The search performance will be gauged on the algorithm's ability to find global optimal solutions in a single run. The experiments will assume that the input parameters affect algorithm performance independently, not taking into account interactions between the input parameters. This is likely not the case, but it is a logical first step towards considering performance factors.

Having generated data for this particular problem, the work of two other authors will be considered. Both have used the GA for optimizing mechanical systems similar to the current work. The perceived strengths and weaknesses of their works will be considered. Of particular interest are the GA input parameters that they've used.

3.3 Performance testing

The input parameters greatly influence the performance of the GA. Detailed studies can be found in the literature relating the input parameter values to algorithm performance. Some published results offer guidelines for appropriate input parameters (Goldberg, 1989b). Few clear trends, however,

String Length (l)	7
Population Size (n)	24
P_{cross}	65%
$P_{mutation}$	0.01%
Maximum Generations	80
Convergence Criteria	$F_{avg} \geq 0.98 F_{max}$

Table 3.1: The 'baseline' GA input parameters for the one variable optimization experiment.

are evident (Grefenstette, 1986). For this reason, the influence of the input parameters on general algorithm performance will be examined in the hope that trends can be found to serve as a guide in future problem formulations.

The effects of varying the individual string length, population size, crossover probability and mutation probability will be compared with the probability of consistently finding the optimal solutions. The previous one variable optimization problem will serve as the test bed. A set of parameters that should yield good results was specified based on recommendations given in Goldberg (1989a). The parameters were then tweaked by trial and error testing to ensure consistency for this particular problem. Table 3.1 shows these baseline input parameters.

3.3.1 Individual string length

Each variable to be optimized is represented as a sequence of binary numbers. For problems having many variables, the individual is represented as one string composed of smaller substrings associated with each variable, as presented in Table 2.1 of Chapter 2. Together with the limits specified for each variable, the substring length determines the variable increment. Larger substrings yield more possible values for each variable, resulting in a finer variable discretization and hence a larger search space.

Given a string composed of m variables each of substring length l, the total string length is $L = l \times m$. The substrings don't have to be of equal length, so one could substring could represent a parameter having two possible values ($l = 1$) and another could represent a substring having 256 possible values ($l = 8$). A total string length of L means that the number of potential solutions in the search space is $2^L = 2^{m \times l}$. Hence an experiment having 4 bits per variable and just one variable is searching through a region of $2^{1\times4} = 16$ possible solutions. A similar experiment having 4 bits per variable and two variables 'sees' a landscape composed of $2^{2\times4} = 256$ possible solutions. The substring length for each variable therefore directly influences the size of the solution space.

Both the number of variables and the substring length are generally specified by the problem definition. For a general design problem there are a collection of candidate factors, or design variables, that can be considered. Each design variable must have an upper and lower bound specified. The substring length for each variable is chosen to realize a suitable variable discretization, hence the number of variables and the substring length are not considered to be variables that can be changed to affect algorithm performance. Rather, they are part of the problem specification and must be understood.

3.3.2 Population size

Population size greatly affects GA search performance. The population must be large enough to adequately search the solution space, while not hindering the progress of the algorithm. Populations that are too small do not sample a large enough portion of the solution space, tending to converge on suboptimal solutions. Populations that are too large hinder the mating of high fitness individuals required to produce higher quality offspring. As a result, these populations converge very slowly and are therefore computationally expensive.

The present study uses a 'baseline' population size of 24 that yields good performance. This was found by simple experimentation. To study the population size effects, experiments will be run with population having from 10 to 60 individuals in 2 individual increments. For each size the algorithm will be

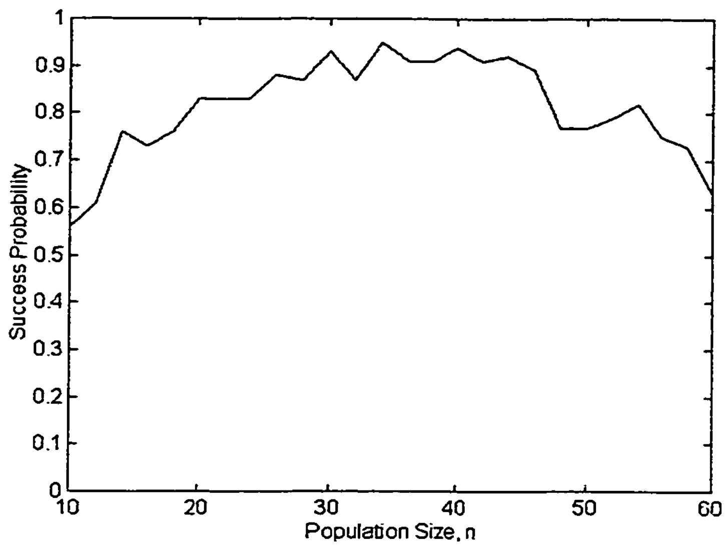

Figure 3.1: Algorithm performance versus population size. Using $P_{cross} = 65\%$, $P_{mutation} = 0.01\%$.

run until completion 100 times. Those runs converging to the known global optimum solution will be considered successful. Those hitting the maximum number of generations or converging to a local maximum are considered failures. It is expected that the probability of convergence versus population size will peak and then tail off in the chosen range.

Results of the population size variation experiment are shown graphically in Figure 3.1. The number of times that the algorithm converged to the global optimum solution out of 100 runs is considered to be the probability of the algorithm converging to the global optimum in a single run. As expected, increasing the population size increases the probability of converging on the global best solution to a point, then falls off. Populations of between 34 to 44 individuals yield the best probability (91% to 95%) of converging on the optimal solution in a single run. Population sizes as small as 20 individuals have a high probability (83%) too, for fewer function evaluations per generation required.

This simple test doesn't show a dramatic effect of population size on algorithm performance. It does indicate that increasing population size until a point steadily increases search performance. After this point algorithm performance drops off. The best conclusion is that the algorithm shouldn't be run just once to find solutions. In practice the GA should be run several times, the optimal solution being the best individual found in all of the runs. A wise strategy might therefore be to run the GA several times with a slightly smaller population size.

3.3.3 Crossover probability, P_{cross}

Crossover probability determines how often genetic material is broken up and recombined when parents mate to produce children. Crossover explores new portions of the solution space by producing new combinations of existing genetic material.

The effect of crossover on the children being produced is dependent on the similarity of the parents' strings. Swapping the heads and tails of two identical parents produces two identical children. This means that two dissimilar yet strong individuals do not necessarily produce strong offspring. Similarly, two poor quality individuals could produce high quality offspring. In mature populations, strong individuals are likely to be combined with one another, thereby dominating the genetic makeup of future populations. Like population size, few guidelines exist for specifying crossover probability.

To gauge the effect of P_{cross} on algorithm performance, P_{cross} will be varied from 0.0 to 1.0 in increments of 0.05. For each value, the algorithm will be run until completion 100 times. As with the population size tests, those runs converging on the optimal solution within the maximum number of generations are considered successful, all others are failures.

It is expected that some crossover is needed as a GA with no crossover can produce (and thereby explore) new solutions to the problem by mutation alone. Thus, except for the occasional mutation, the algorithm is left to repeatedly select mates from a population identical to the initial population,

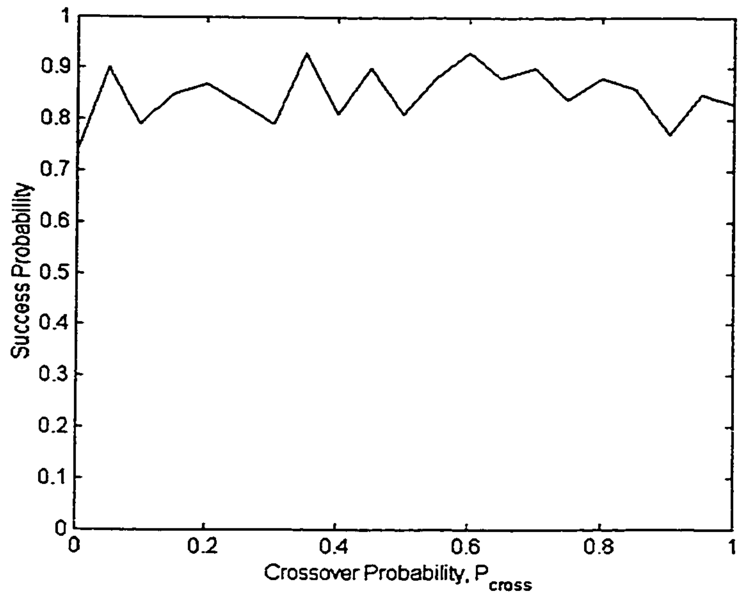

Figure 3.2: Algorithm performance versus crossover probability. Using $n = 24$, $P_{mutation} = 0.01\%$

possibly converging on the best individual of this population by slowly excluding all of the others having lower fitness from being selected as mates. A crossover probability of 1.0 means that all unions between mates produce offspring by combining the parents' genetic material. This means that no direct copies of individuals get passed to successive generations, hence strong individuals only retain their superior makeup if they're combined with identical mates.

Figure 3.2 graphs the results of the P_{cross} variation test. As in the previous experiment, the number of times that the algorithm converged to the global optimum solution out of 100 runs is considered the probability of the algorithm converging to the global optimum in a single run. Surprisingly, a significant trend is not evident with different values of P_{cross}.

The probability of converging on the optimal solution in a single run of the GA ranged from 74% (where $P_{cross} = 0\%$) to 93% (where $P_{cross} = 60\%$).

This experiment indicates that running the GA with no crossover isn't too bad of an idea, yielding success about three-quarters of the time. This result is interesting, but doubtful: a rate of 0% couldn't possibly yield an aggressive search likely required to explore new territories in a real problem. Its probable that the test was too simple, having few local maxima and a large population with respect to the search space.

The results indicate that the value of P_{cross} does affect algorithm performance. The slightly higher performance scores around the 60% mark reinforce, but don't prove, the notion that a moderate crossover rate yields acceptable performance.

3.3.4 Mutation probability, $P_{mutation}$

Mutation probability determines how often genetic material is randomly disturbed when parents produce offspring. This process explores new areas of the solution space in a purely random manner. The effect of mutation on an individual is dependent on where the mutation occurs in the individual's bit string: mutation in higher order bits of the individual's substrings can significantly change the individual's fitness. Mutation in less significant order bits may not change fitness significantly.

The standard references (Goldberg, 1989a) note that some mutation is required to promote exploration of new solution regions, yet too much mutation becomes disruptive, hampering algorithm convergence. Values of $P_{mutation}$ from 0.01% to 2% are typical (Roston and Sturges, 1996; Kim, 1995; Segla *et al.*, 1998). The mutation study will therefore alter $P_{mutation}$ from 0% to 2% in 0.1% increments.

Figure 3.3 plots the $P_{mutation}$ variation test. As in the other experiments, the number of times that the algorithm converged to the global optimum solution out of 100 runs is considered the probability of the algorithm converging to the global optimum in a single run. Like the crossover probability experiment, discerning trends is difficult. The probability of converging on the global optimal solution in a single run of the GA ranged from 66% with

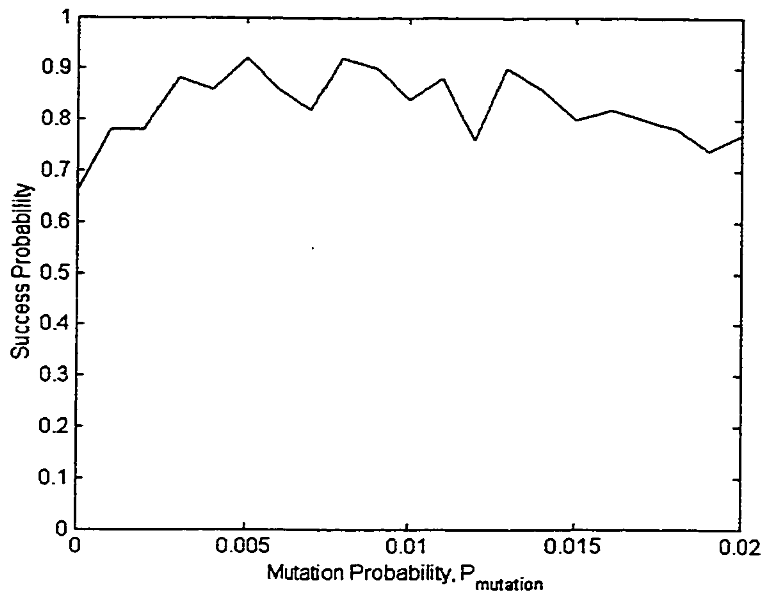

Figure 3.3: Algorithm performance versus mutation probability. Using $n = 24$, $P_{cross} = 65\%$

no mutation to 92% with a 0.8% mutation rate.

As with the crossover probability rate tests, this experiment shows that there is a variance of algorithm performance with the mutation rate. A small mutation rate, slightly less than 1%, appears wise from this simple test.

3.4 Mechanical design applications

This section will consider previous applications of the GA to solving a particular type of mechanical design problem known as path generation synthesis. Path generation synthesis problems involve finding the appropriate mechanism dimensions such that a point on one of the links of the mechanism passes through several prescribed locations. In both of the works considered the GA is being used to find the optimal combination of link lengths of a four bar linkage such that a point on the coupler link of the four bar passes

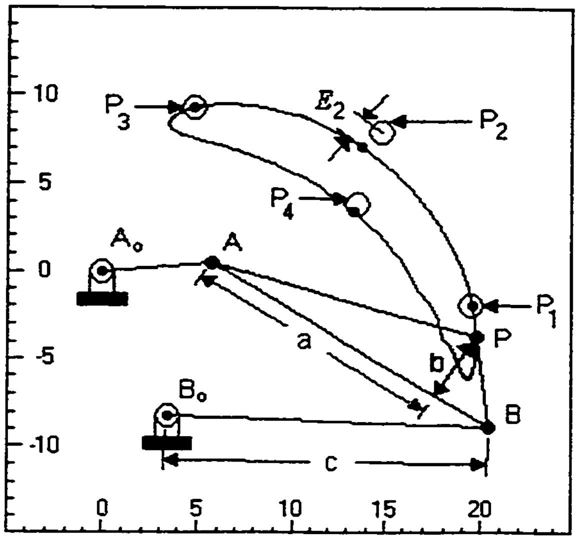

Figure 3.4: Four bar linkage with fixed pivots A_o, B_o, moving pivots A, B and precision point P. Points P_i are specified points, E_2 is tracking error for second prescribed point.

through the set of target points with a minimum tracking error. A typical four bar linkage and its coupler curve is shown in Figure 3.4, plotted over a set of prescribed target points. The shortest distance, e_i, between each target point and the coupler curve is known as the tracking error for the ith point.

3.4.1 Path generation synthesis - Roston and Sturges

Roston and Sturges (1996) investigated using the GA as an optimization tool in the design of planar four bar linkages passing through prescribed point sets with a minimum tracking error. They considered only those mechanisms that were of Grashof type, or those four bars in which one link could be driven through a complete revolution. The goal of the experiment was to find an appropriate objective function for the GA to minimize that would result in high quality GA solutions representing those mechanisms that pass through a specified point set with a low tolerance.

Specified points formed by:	Number of specified points 3 or 4	5	8	11 or 24
Four-bar mechanism	$\surd$	$\surd$	$\surd$	$\surd$
Involute Curve	$\surd$	$\surd$	$\surd$	X
Straight Line	$\surd$	X	X	$\surd$

Table 3.2: The experiments and the number of points for each experiment from Roston and Sturges (1996).

A series of design tests was performed using the GA. The dimensions uniquely specifying a four bar linkage comprised the set of variables to be optimized as the number of prescribed target points ranged from 3 to 24. These prescribed points were taken from three sources: four bar linkages with known geometry in the solution space, an involute curve and a straight line. Table 3.2 specifies the experiments performed for each source of prescribed points, a $\surd$ meaning the curve source - number of precision points set was tested, a X meaning it wasn't.

A simple GA was used that employed the concepts of *elitism* and *population decimation*. Elitism entails simply copying several of the best individuals from the previous generation to the next generation. This ensures that the best individuals are always present in future populations, promoting search in the best areas found. The population decimation method creates a much larger initial random population than the working population size. From this large initial pool of random individuals, the top n are chosen to create the first population, where n is the working population size. In this manner a larger portion of the solution space is initially examined and only the fittest individuals are selected to the first population.

The GA input parameters for this experiment are shown in Table 3.3. These input parameters were chosen not according to a specific methodology, but based on recommendations in Goldberg (1989a). The space being searched by this experiment was massive, each of the nine input parameters

Substring length (l)	16
Number of parameters (m)	9
Total string length (L)	$16 \times 9 = 144$
Population size (n)	200
P_{cross}	50%
$P_{mutation}$	1.0%
Number of elitists / generation	10
Maximum generations	150
Convergence criteria	150 generations

Table 3.3: The GA input parameters used in Roston and Sturges (1996).

being represented by 16 bits for an overall string length of 144 bits. The search therefore considered $2^{16\times 9} = 2^{144} = 2.23 \times 10^{43}$ possible distinct solutions. In relation to this large string length and hence search space, the population size of 200 is quite small.

The input length variables were limited to the range 0.0 to 10.0 units; the input angular variables ranged from 0.0 to 2π radians. This yielded a very fine length input discretization of

$$\delta_l = \frac{10-0}{2^{16}-1} = 0.0001526$$

units per increment and a similarly fine angular input discretization of

$$\delta_\theta = \frac{2\pi - 0}{2^{16}-1} = 0.0000959$$

radians per increment. No convergence criteria utilizing population fitness values was employed; the GA was always run for 150 generations, the best individuals found in any generation being the possible solutions to the problem.

Roston and Sturges' fitness function was of the form

$$f = \sum_{i=0}^{m} \begin{cases} 1/C & e_i \leq C \\ 1/e_i & e_i > C \end{cases}, \tag{3.1}$$

where e_i is the tracking error associated with the ith member in the set of m prescribed points. The constant C is a factor that is dynamically reduced "once a user-specified number of mechanisms satisfy $f_i = (m)/C$". This fitness function correctly rewards high fitness scores for those mechanisms whose paths pass near to the prescribed point set and poor scores to those that do not. The dynamic reduction of the factor C limits the amount that a single good tracking error in the set influences the fitness assessment. This function therefore correctly rewards the highest fitness scores to solutions having small tracking errors throughout the prescribed point set and not at just a single location. This will be considered further.

Roston and Sturges found that the experiments having few prescribed points (five or less) led to many different solutions exhibiting small tracking errors. Very few of those solutions found, however, resembled the known mechanism used to generate the input point set or had paths resembling that of the known solution. As the number of points prescribed was increased, the good solutions found still did not resemble the known input solution, but the generated curves looked more similar to the input data curves. This behaviour seems reasonable as few prescribed points impose fewer constraints on the design problem. It was therefore found that to find mechanisms having paths that are similar to the known input curve, many points must be specified thereby constraining the design problem.

The quality of the solutions found was measured in terms of the maximum path tracking error for a solution. Maximum tracking errors ranged from 0.01 to 0.31 units. In terms of the length scale discretization, $\delta_l = 0.0001526$, these tracking errors are quite large. Comparing the tracking error to the maximum possible value of each length variable ($length_{max} = 10$) indicates that the mechanisms found had tracking errors comprising from 0.1% to 3.1% of the input variable maximum.

Roston and Sturges' study highlights the feasibility of the GA as a tool for designing path generating mechanisms. As no study was undertaken to validate the GA input parameters, future studies should be concerned with finding a set of input parameters that yields consistent results when tested using several known solutions. This would ensure that the input parameters yielded optimal or near optimal search performance, and thereby validate the results found in real design task experiments.

Though no results were presented, Roston and Sturges maintain that the fitness function used in their study promotes convergence of the GA to global solutions. Formulating a fitness function based on the inverse of the summation of the tracking errors or the summation of the inverses of the tracking errors leads to local convergence. The use of a dynamically changing factor ensures that a single close prescribed point does not dominate the individual fitness assessment.

The investigators note that a method is needed to quantify the ability of the mechanism in satisfying the design task. In order to compare two mechanisms, A and B, on two different design tasks (prescribed point sets), a method is needed for assigning a score to a mechanism that takes into account the path tracking errors, the overall mechanism size and the input variable discretization. Comparing the tracking error to the sum of the mechanisms' link lengths, the input variable discretization or the maximum distance between prescribed points are possible ways to yield a normalized measure of the solution quality.

3.4.2 Path generation synthesis - Kim

As in Roston and Sturges (1996), Kim (1995) investigated using a genetic algorithm as a tool for designing planar four bar linkages. Kim combined a simple GA with a gradient based local search technique, deeming the combination a hybrid GA. The purpose of Kim's study was to find a method that was relatively fast and could obtain highly accurate solutions for determining the geometry of four bar linkages passing nearly through sets of specified precision points. Only the design of Grashof four bar linkages, or those four bars in which one link can be driven through a complete revolution, were considered.

Kim performed two main experiments. The first experiment attempted to find an optimal set of GA input parameters that would result in global optimum solutions to several known problems in a minimum number of function evaluations. This hybrid GA with optimal input parameters was then compared to a random search method combined with the same local gradient based search technique. The second test used the hybrid GA to find optimal linkage lengths of path generating four bar linkages. Four path generation design tasks with known solutions were tested.

Kim's hybrid GA used a local gradient based search technique to explore the regions represented by the top k best individuals of each population in the GA. The local search method was initialized with each of these top k solutions and run until completion. The result of this search represented the best solution in the vicinity of each of the top individuals. If this solution did not have a path tracking error less than a specified amount, the GA individual that provided the starting point was thrown out of the population. It was theorized that removing all of these sub optimal individuals eliminated potential sources of local convergence for the GA, thereby increasing the chance of finding the global optimum solution. If the solution found by the local search technique had a path tracking error less than the specified amount, this individual was deemed an excellent solution to the problem and the hybrid GA searching process was ended.

The simple GA component of Kim's hybrid scheme used a roulette wheel selection method, one point crossover and simple mutation. The GA input parameters for these experiments are shown in Table 3.4. As in Roston and Sturges (1996), nine variables completely specifying a four bar linkage were being optimized by the hybrid GA. The limits on the design variables and the resulting input variable discretization are shown in Table 3.5. The substrings representing each variable were of equal length and were varied from 6 to 10 bits per parameter. The size of the region being searched therefore ranged from $2^{6\times 9} = 1.80 \times 10^{16}$ to $2^{10\times 9} = 1.24 \times 10^{27}$ possible solutions. Goldberg's theoretical estimate (Goldberg, 1989b) calls for a population size of approxi-

Substring length (l)	$6 \rightarrow 10$
Number of parameters (m)	9
Total string length (L)	$(6 \rightarrow 10) \times 9 = 54 \rightarrow 90$
Population size (n)	100
P_{cross}	60%
$P_{mutation}$	7%
Maximum generations	150
Convergence criteria	Solution with $\sum_{i=0}^{m} e_i \leq 0.001$

Table 3.4: The GA input parameters used in Kim.

Input Variable	Range	Discretization (6 bits $\rightarrow$ 10 bits)
Base point location, A_o	$-2.0 \rightarrow 2.0$	$0.0635 \rightarrow 0.0039$
Link lengths, l_i	$0.01 \rightarrow 2.0$	$0.0316 \rightarrow 0.0019$
Location of P on coupler	$-2.0 \rightarrow 2.0$	$0.0635 \rightarrow 0.0039$

Table 3.5: The variable limits and resulting discretizations for Kim's path generation synthesis experiments.

mately 4300 when an overall string length of 54 is used. While successful GA searches have used far smaller population sizes than Goldberg's theoretical estimate, a population of 100 seems small for such large individual strings.

Kim did not define a fitness based convergence criteria for search termination. Instead, the hybrid GA was run until a single individual was found that had a sufficiently small tracking error (this will be discussed shortly) or the GA had undergone the maximum of 150 generations. This means that Kim's hybrid GA searched for a single best solution to the problem. Using some form of convergence criteria based on population fitness values means that the GA terminates when several high fitness solutions have been found for the problem. While it may seem redundant to continue searching after a high quality solution has been found, other, different, high quality solutions may be subsequently found. This could potentially result in several excellent solutions being found for the problem, giving the mechanism designer several alternative designs from which to choose a best one.

The limits placed on each variable are shown in Table 3.5. Shown also is the resulting discretization for each variable when the variable substrings are represented as bit strings of lengths 6 and 10. This yielded an input variable discretization ranging from 0.0316 to 0.0635 for a 6 bit string to 0.0019 to 0.0039 for a 10 bit string.

The fitness function used to assess an individual was of the form

$$f = \sum_{i=0}^{m} e_i, \tag{3.2}$$

where e_i is the tracking error between a prescribed point and its associated closest curve point. This simple summation of point tracking errors yields a low score for mechanisms passing nearly through all of the prescribed points and a high score for those mechanisms that generate coupler curves that do not pass through the prescribed points. The method used to transform small tracking error summations to high fitness values is unclear; Kim refers to those individuals that pass nearly through the prescribed locations as having both high fitness and a small tracking error summation, yet no relationship is presented that relates small errors to high fitness values.

Kim's performance testing experiments were built upon varying the other GA input parameters with respect to mutation probability, $P_{mutation}$. It was found that "in some range of mutation probability (between 0.2% and 0.8%), other parameters do not have an effect on the algorithm's performance". This result is not unexpected; in an earlier study Grefenstette (1986) asserts that "mutation rates above 5% are generally harmful to algorithm performance and rates greater than 10% yield GA performance approaching that of random search, regardless of other parameter settings". Due to the high mutation rates considered, no trends can be seen in the variation of other parameters with respect to the algorithm's searching performance. It is interesting to note that for Kim's subsequent design experiments that a mutation rate of 0.07% was used.

Results of the comparison between the hybrid GA and the hybrid random search showed that the hybrid GA found the best solutions to known problems in approximately half of the function evaluations required by the hybrid random search. Both hybrid search methods were run on the same problem, that of finding the correct geometry of a known four bar linkage producing a coupler curve passing through a set of design points. The prescribed point set consisted of nine points taken at random locations from the known solution's coupler curve. The hybrid GA, finding the optimal solution in approximately half of the function evaluations, was shown to be the superior search method.

Kim's design problems were taken from Wampler *et al.* (1992). Wampler showed that for the problem of nine prescribed points for a four bar linkage to pass through, there are generically 1442 non-degenerate solutions along with their Roberts cognates, for a total of 4326 distinct mechanisms. Most of these 1442 solutions, however, have complex link lengths. Those solutions representing actual physical mechanisms were mostly subject to branch (the mechanism must be disassembled then reassembled at some point in its range of mobility) or order (the mechanism does not pass through the point set in the prescribed order) defects.

Wampler considered four problems having nine prescribed points. The first problem was found to have 21 possible physical solutions, few of which did not have branch or order defects. Wampler identified several of these that did not have branch or order defects. Wampler's second problem had 45 physical solutions, all having branch or order defects. This problem therefore is physically impossible to solve. The third problem of Wampler had 64 physical solutions, only one of which did not have branch or order defects. The last problem of Wampler, problem 4, had nine points taken from an ellipse for a total of 120 possible mechanisms. It is unclear how many of these 120 mechanisms did not have branch or order defects.

Kim's results were in agreement with the results of Wampler's experiments. Kim found proper solutions matching those of Wampler in 15 of 25 runs of the GA for problem 1. Problem 2 didn't yield any solutions having acceptable tracking errors, as expected. Kim found the single solution to problem 3 in each of 50 runs of the GA. Problem 4 did not produce very good results; Kim found possible solutions to the problem in 6 of 25 runs of the GA. Based on these results, Kim was able to show that the hybrid GA could be used to find highly accurate solutions in a very large, highly non linear solution space.

As noted in Roston and Sturges (1996), using the sum of the individual tracking errors as the basis of a fitness function tends to promote local convergence in the simple GA. The local search component of Kim's hybrid GA theoretically eliminates these sources of premature convergence and Kim's high accuracy solutions seem to support this. It would be interesting, however, to test the hybrid GA scheme on the same experiments using a fitness function that exhibits better global search capabilities when implemented in the simple GA alone.

The population size used by Kim was small, being far less than what is suggested by Goldberg (1989b). While Goldberg's estimate for population size in terms of string length is so high that it's impractical, Kim makes no mention of a criteria for choosing population size. In most GA implementations, a suitable population size is one that consistently yields the optimal solution to a known problem that is similar to the problem to be tested.

Kim's convergence criteria, while economical in terms of the number of function evaluations performed, finds a single solution per run to the problem. A more common population fitness based convergence criteria may yield several good solutions to the problem by not terminating upon finding the first successful solution. This would let the designer choose from several possible solutions to the problem.

3.5 GA Conclusions

In order to realize optimal solutions using the GA, a proper input parameter set must be used. An appropriate set of input parameters can be identified as a set that yields consistent optimal results on a known problem similar to the experimental problem. Therefore if a set of input parameters results in the GA consistently yielding a small number of high quality solutions, one of which is the known global best solution, then this is the input parameter set that should be used for further searching on similar problems. Values of P_{cross} between 50% and 90% are typical; mutation rates less than 2% have been shown to yield good results with the previous P_{cross} range.

Probably the most important parameter, population size, is often the most difficult to specify. Too small a population and the GA will tend to converge on local solutions, too large and many generations will be required to refine a population so that it converges on a global best solution. For a given string length, the GA should be run repeatedly until completion using progressively larger population sizes until one is found that repeatably finds global optimal solutions. When an appropriate input parameter set has been found the GA should be run several times on each experiment. If the GA finds a high quality solution several times throughout the runs, it can be assumed that this is the global optimal solution.

Great care must be taken in formulating a fitness function to ensure that the function represents the quantity that the designer wishes to maximize. In both of the path generation studies, the goal was to find a mechanism having the smallest tracking error passing near to a set of prescribed target points. Both investigators used a fitness function based on the summation of the tracking errors. Roston and Sturges identified that either summing the reciprocals of the errors or taking the reciprocal of the error sum promoted convergence on suboptimal solutions. Kim used a sum of the tracking errors as his fitness function, using a local search technique to find suboptimal solutions and remove them from the population. Both studies could perhaps have avoided these problems by basing their fitness functions on the largest tracking error in the set of errors. This would mean that the best solutions identified by the GA would have the smallest tracking errors, both at the individual points and the sum of these individual errors. Hence several different fitness functions should be tested for a given problem to see if similar solutions are found with each one.

Chapter 4

Four bar linkage path generation synthesis using the GA

4.1 Chapter Glossary

- a - x- position of P in coupler's body fixed coordinate system.
- b - y- position of P in coupler's body fixed coordinate system.
- c - length of follower link.
- P_i - a discrete path point generated by a solution.
- T_i - a prescribed point.
- m - the number of prescribed points.
- n - the population size.
- Q - the closest continuous path point to a prescribed point.
- L_c - the closest distance between a continuous curve and a prescribed point.

- L_{cd} - the closest distance between a discrete path point and a prescribed point.
- e_i - the tracking error of prescribed point i with respect to the curve.
- E - the set of tracking errors e_i.
- $Max(E)$ - the maximum tracking error in the set E.
- e_{max} - $= Max(E)$.
- A_o, B_o - the four bar linkage fixed pivot point locations
- A, B - the four bar linkage moving pivot point locations
- P - the precision point on the four bar linkage
- $P_{crossover}$ - the crossover probability for the GA
- $P_{mutation}$ - the mutation probability for the GA
- δ - the GA input variable increment
- Objective - the quantity to be minimized by the GA.
- Fitness - the GA-scaled objective value.

4.2 Introduction and Background

A mechanism is defined (Mallik *et al.* (1994)) as a collection of rigid bodies that move with definite relative motion so as to transfer motion from a source to an output. Mechanisms can be broadly classified as open or closed loop, each link or body of a closed loop mechanism being connected to at least two other links. The simplest closed loop mechanism is the planar four bar linkage, shown in Figure 4.1. It is composed of four rigid bodies (bars) and four revolute joints, forming a closed loop kinematic chain. This class of mechanism has been used to solve numerous design tasks due to its many variations and simplicity.

Figure 4.2 shows some typical applications of four bar linkages that can be broadly categorized based on their kinematic tasks (Sandor and Erdman (1984)). Four bar linkages intended to transfer an input motion to an output motion are referred to as *function* generators. Those whose primary task is to guide a fixed point on one of the links along a path are termed *path* generators. *Motion* generators differ from path generators in that a rigid body is guided through several locations in the plane. The present study will be concerned primarily with the design of path generating mechanisms.

Planar four bar mechanisms can be further classified according to their range of movement. Those mechanisms in which the shortest link can make a complete revolution with respect to all of the other links are said to be of *Grashof* type. Grashof types satisfy the condition

$$s + l < p + q \tag{4.1}$$

where s, l, p, q are the shortest, longest and other two link lengths, respectively. Proof of the Grashof criteria and further details are found in Mallik *et al.* (1994). The four Grashof types and their naming conventions are shown in Figure 4.3. All of these mechanisms can accept a rotating input.

Figure 4.4 shows a crank-rocker mechanism passing through four prescribed locations. Designing four bar mechanisms that pass through a set

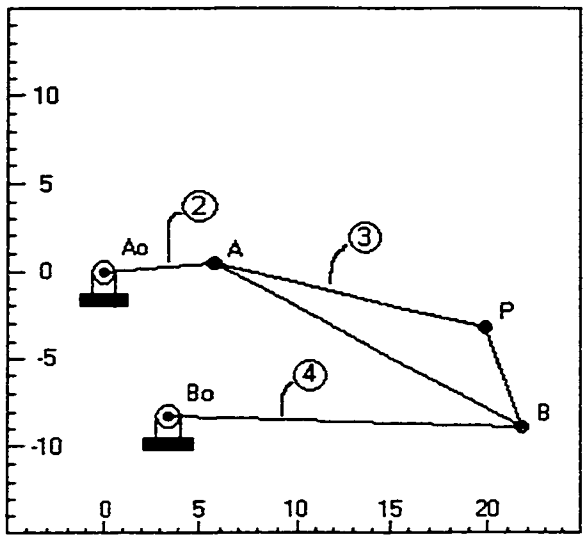

Figure 4.1: The four bar linkage. Link 2 is the crank, link 4 is the follower and link 3 is the coupler. Link 1 is the grounded body that connects the moving links at ground points A_0 and B_0. P is the path tracer point.

of specified points is known as path generation synthesis. The mechanism is sought whose geometry is such that the design point will pass through the given precision point set in the proper order.

Path generation synthesis can be performed using exact or approximate techniques. Exact synthesis methods attempt to find the mechanism whose path passes perfectly through the prescribed precision points. Exact methods can be implemented graphically or analytically (Sandor and Erdman (1984), Mallik *et al.* (1994)) and are limited to, save for very special cases, a maximum of five specified precision points.

Approximate synthesis methods try to determine the mechanisms whose paths pass within some tolerance of the prescribed precision point set. Approximate methods are therefore a form of optimization problem where a minimum tracking error is the quantity being sought. The number of precision points that can be prescribed for approximate synthesis methods is theoretically infinite, being limited practically by the path error tolerance.

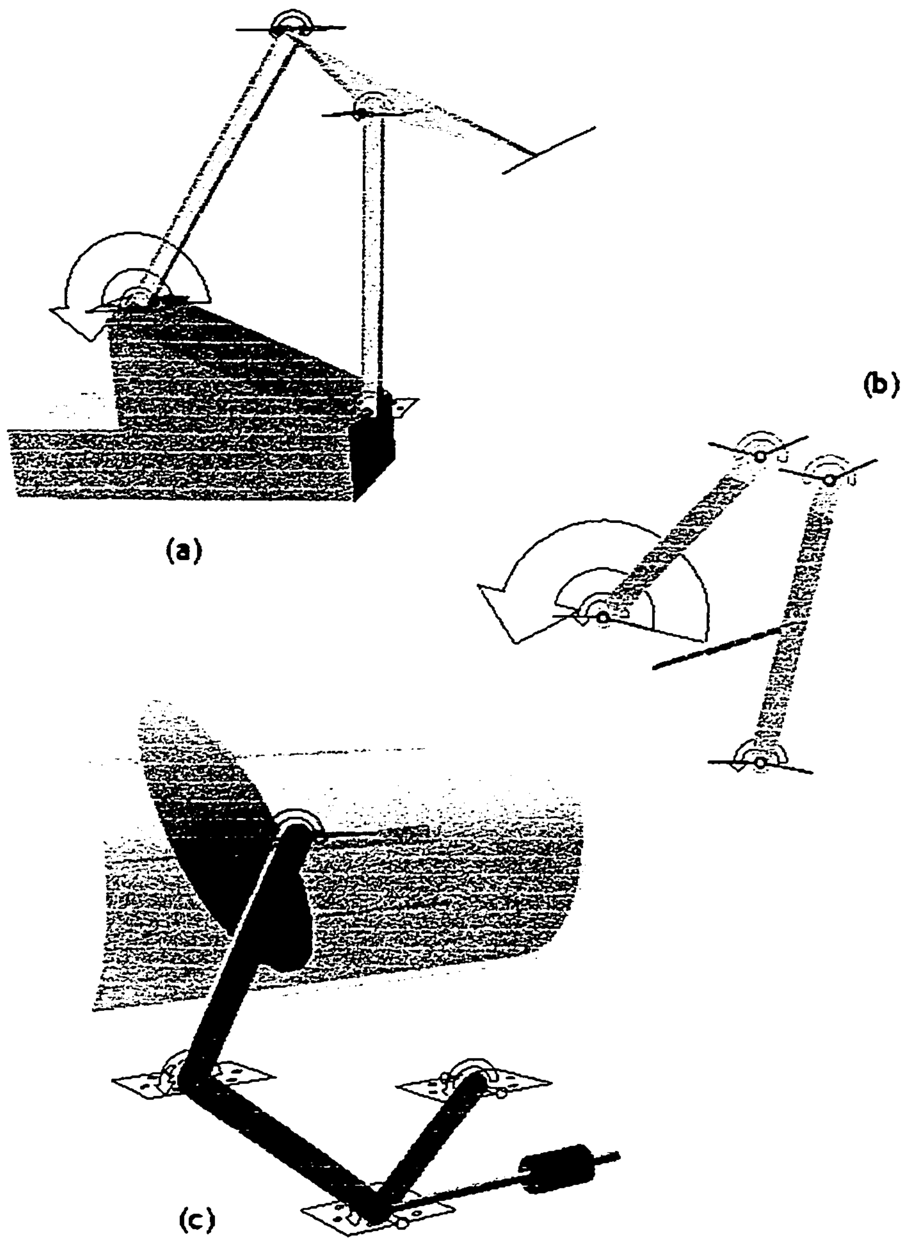

Figure 4.2: Four bars and their kinematic tasks: (a) path generation (b) motion generation and (c) function generation.

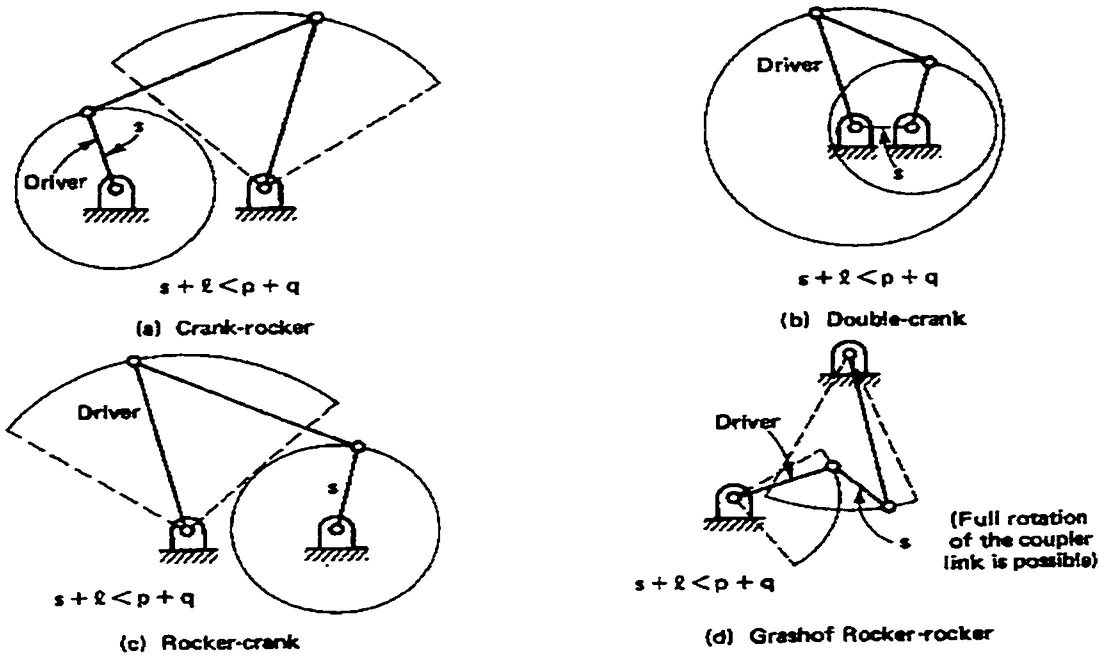

Figure 4.3: The four Grashof types. The shortest link of each mechanism is capable of undergoing a complete revolution (taken from Sandor and Erdman (1984))

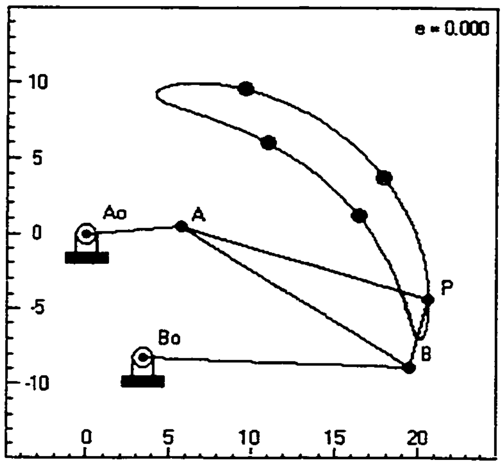

Figure 4.4: A four bar linkage passing through four prescribed locations. The design point, P, on this mechanism passes through the prescribed point set with zero tracking error.

4.3 Problem definition

Path generation synthesis attempts to find a mechanism which generates a path passing through several prescribed points. If some tolerance is allowable in the path then the design task can be approached using a numerical optimization technique. This amounts to a multi-variable optimization problem where the variables to be optimized represent the mechanism geometry and the quantity to be minimized is the path tracking error.

Any numerical optimization technique requires, at minimum, an objective function to minimize or maximize. This mathematical representation of solution quality awards high scores to those solutions having desirable characteristics and low scores to those having poor characteristics. Most design objectives can be expressed in a variety of mathematical forms, some depict better reflections of the desired optimization criteria than others. As we seek the mechanisms that trace paths passing as close to the precision point set as possible, it is clear that the prescribed point set tracking errors represent the quantities to be minimized. An appropriate mathematical depiction therefore must be found.

Having specified a mathematical representation of the desirable solutions' qualities, proper inputs are needed for the GA which is used to solve this optimization problem. Once a set of input parameters are found for a GA searching a certain sized landscape, these inputs typically yield acceptable performance on different problems having similar search scales. This problem independence is an attractive quality, yet care must be taken to ensure that a chosen input parameter set truly does result regularly in finding optimal solutions. In particular, a suitable population size must be used. Small populations typically have difficulty finding optimal solutions and large populations take many generations to converge, thereby prolonging the search times.

As in analytical four bar design methods, it is expected that the number of points specified in the path generation problem influences the possibility of finding an excellent solution. Problems having few prescribed points will likely have many solutions in the search space. As more precision points place more constraints on the design problem, the number of possible mechanisms in the search space satisfying the design criteria decreases. The effect of the number and arrangement of prescribed points will be considered.

Once the GA has converged on a solution to the problem, a method is needed to assess the quality of this solution in terms of the problem definition. Often the designer has a maximum acceptable path tracking error in mind when the problem is formulated. If this is not the case, it may be useful to specify the path tracking error in terms of a typical link in the mechanism. For example, a mechanism having a maximum path tracking error of 0.01 units may be a good solution when the mechanism's smallest link length is 10 units. This may not be such a good solution when the smallest link is 0.1 units long. The designer can assess the solution quality by comparing the tracking error of the best found solution to the input variable increments, prescribed point spacing and/or mechanism dimensions. A criteria is needed to normalize the path tracking error in terms of one or more of these quanti-

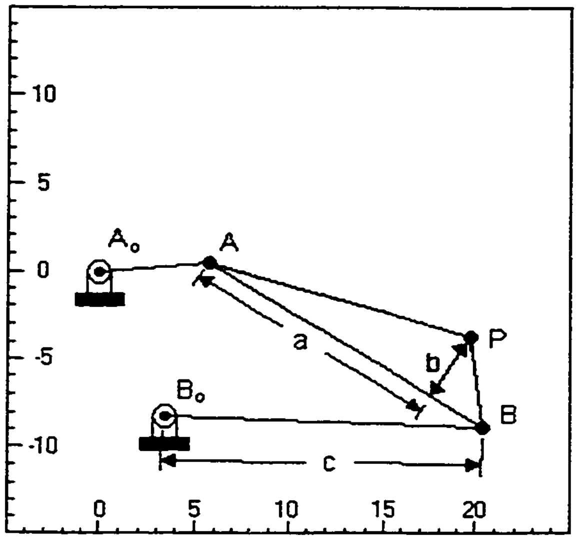

Figure 4.5: Variables a, b and c size the follower link and locate point P on the coupler link.

ties, thereby enabling some form of comparison between solution quality for different problems.

4.4 Method

All problems considered here involve finding the appropriate geometry of a four bar linkage that will result in a coupler path passing nearly through the prescribed point set. We will consider only three variables of the four bar linkage in our optimization problem. Shown in Figure 4.5, the variables a, b, c size the follower link B_oB and locate the path tracer P on the coupler link. The location of the fixed points A_o and B_o and the length of the driver and coupler are assumed given.

All of the mechanisms considered in this study will be Grashof crank-rocker types; the follower link length variable limits do not allow for any other four bar linkage type. The positions of the coupler and follower links are found using the Newton-Raphson iterative root finding technique (Appendix A) as the crank is driven through a complete revolution in 1^0 increments.

In order to gauge the solution quality, the notion of 'good' and 'excellent' solutions will be defined in terms of the finest input variable increment. This is logical as the designer has knowledge of, and control over, the input parameters and their discretizations. Those solutions having a maximum tracking error, E_{max}, less than the finest input increment, δ (where $\delta = (10 - 2)/(2^8 - 1) = 0.0314$ for the input variable with bounds 2 and 10 that is discretized into 256 levels), are deemed 'excellent' solutions. Those having E_{max} less than double ($2\delta = 2 \times (10 - 2)/(2^8 - 1) = 0.0628$) the finest input increment are deemed 'good'. The final experiments will examine parameterizing the maximum tracking error in terms of other problem specific quantities.

4.4.1 Finding appropriate GA inputs

In order to effectively search the solution space, an appropriate GA input parameter set must be used. Due to the high problem independence of the GA search performance with respect to the input parameter set, a parameter set that works well on one search will generally work well on others having similarly sized landscapes. As found previously (section 2.3), an appropriate population size appears to be the most important factor in achieving successful search performance. We will consider only variations in population size as it was shown earlier that a high P_{cross} rate and a small $P_{mutation}$ rate yield good search performance.

In order to test the searching effectiveness of a particular population size, the GA will be run several times with different population sizes on a problem having a known solution contained in the search space. The number of times that the algorithm converges to the known best solution over several runs of the GA will be considered the indicator of the input parameter set suitability.

The known solution for these tests will have many prescribed points. This is done in order to place many constraints on the problem, thereby minimizing

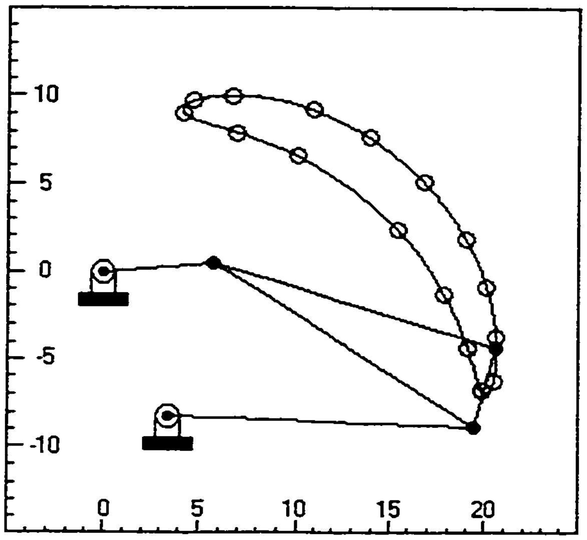

Figure 4.6: Sixteen prescribed points (hollow circles), taken from the path generated by the four bar linkage having $a = 15.0$, $b = 4.4$ and $c = 16.0$.

the number of possible excellent solutions within the search space. A set of 16 target points, shown in Figure 4.6, was chosen arbitrarily from the path of the known mechanism having $a = 15.0$, $b = 4.4$ and $c = 16.0$ to be the prescribed point set.

The GA input parameters are shown in Table 4.1. The search variable limits and the resulting variable discretizations are shown in Table 4.2. Each variable will be represented by 8 bits per variable, hence there are 256 possible values for each including the upper and lower limits. The population size is varied from 60 to 200 individuals in increments of 20. The GA is run until completion for each population size 20 times. A run is completed when the convergence criteria has been met - in this case the population average fitness is greater than or equal to 98% of the maximum fitness individual ever found - or the maximum number of generations has elapsed. The number of times that the GA finds the known or similar high quality solution is used to calculate the probability of finding a high quality solution in one run of the GA.

Parameter	Value
Population Size	$60 \rightarrow 200$
$P_{Crossover}$	65%
$P_{Mutation}$	0.8%
Bits/Parameter	8
Maximum Generations	200
Convergence Criteria	$0.98 \times F_{max} \leq F_{avg}$

Table 4.1: GA inputs.

Parameter	Lower limit	Upper limit	δ
a	10	20	.0392
b	2	10	.0314
c	12	20	.0314

Table 4.2: Input variable limits.

4.4.2 Objective function formulation

The mathematical representation, or objective function, of the quantity to be minimized must assign high scores to those mechanisms having paths passing near to the prescribed points and low scores to those not passing near the prescribed points. This means that the tracking error from each prescribed point to the coupler curve must first be found. This set of tracking errors is then used to calculate an objective function value.

Calculating path tracking error

Given a set of discrete locations for the path of P on the coupler of an arbitrary mechanism and m prescribed point locations, the path tracking error, e_i, must be calculated for each prescribed point. Coupler paths not passing exactly through the precision points must be assigned a tracking error value. Those paths passing perfectly through the defined set must be

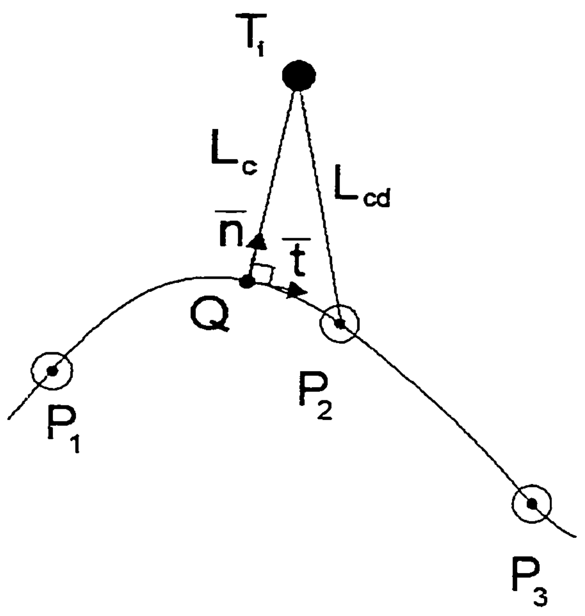

Figure 4.7: Calculating the true path tracking error.

assigned tracking errors of zero. Each precision point will have one or more[1] associated closest points within the coupler point set. A sequential traverse of the path yields the closest discrete path point for each precision point.

Figure 4.7 shows a set of three discrete path points, $P_{1\rightarrow3}$, on a continuous curve and a single prescribed point (precision point), T_i. Assuming that the data points are closely spaced along the curve, a crude estimate of the path tracking error is simply the distance from the closest discrete path point to the precision point, the distance L_{cd}. The true path tracking error can be found when an extension of the unit normal vector to the curve, $\vec{n}$, passes through the precision point. This point on the curve, Q, is the closest to the precision point and the distance L_c is the true tracking error of the continuous curve with respect to the precision point.

To calculate L_c, the discrete path points would have to be fitted with

[1]Note that a minimum separation distance from a precision point may be shared by two or more path points.

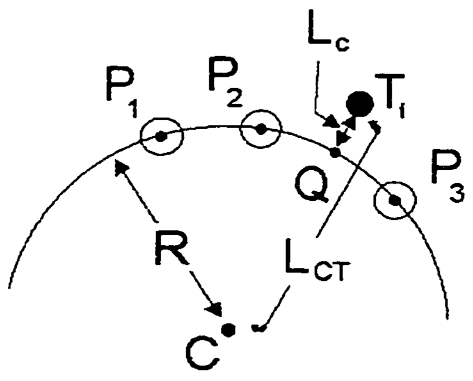

Figure 4.8: Approximating the path tracking error using a circle fit.

differentiable functions in terms of x and y using an interpolation scheme. The interpolated curve could then be traversed incrementally in the region of the closest discrete path point, until a point Q is found at which an extension of $\vec{n}$ passes through T_i, thereby defining L_c.

While this method would yield a highly accurate estimation of the true path tracking error, it is computationally intensive. A less rigorous method is used by Kim (1995) to estimate the path tracking error. Kim fits a circle (see Appendix B) to the three discrete path points represented by the closest point and its adjacent points. Knowing the location of the circle center, C, an estimate of the path tracking error, L_c, can be found by constructing Figure 4.8.

The equation

$$L_c = L_{CT} - R \tag{4.2}$$

estimates the path tracking error in terms of the known circle radius R and the distance, L_{CT}, from the circle center C to the prescribed point T_i. In what follows the path tracking error for point T_i is defined to be e_i, where $e_i = L_c$ is a positive number.

The closest discrete point distance, L_{cd} of Figure 4.7, will be compared to Kim's estimate of the true tracking error, L_c of Figure 4.8, by comparing the calculated tracking error magnitude of points on the continuous curve that do not appear in the discrete point path set. For the known solution geometry ($a = 15.0$, $b = 4.4$ and $c = 16.0$), the crank link will be driven through a complete revolution in 0.5^0 increments. From this set of discrete points, three that are not present in the set found using a 1^o crank link driving increment will be chosen as prescribed points. As these points are on the continuous curve produced by the path tracer point P, it is expected that they will be assigned negligible error magnitudes if they are used as prescribed points for the coarser (1^o) driving increment.

Transforming tracking error into fitness score

Having calculated the set of tracking errors E, where $E = (e_1, e_2, ..., e_m)$ for m prescribed points, a method is needed to transform E into an fitness function value. The GA requires that better solutions to the problem receive higher fitness values, hence a solution with low tracking errors for each precision point should receive a high fitness score and a solution having large tracking errors should receive a low fitness score.

A simple fitness relationship can be found by taking the reciprocal of the sum of the tracking errors:

$$Fitness = (\sum_{i=1}^{m} e_i)^{-1}. \tag{4.3}$$

This scheme, while correctly assigning high fitness to mechanisms having small tracking errors, was found to be unsuitable by Roston and Sturges (1996), as it tends to promote local convergence of the GA. The downfall of this method is illustrated by the two possible solutions to the problem shown in Table 4.3. Solutions Z and W are two possible solutions to a problem having three specified precision points and hence three associated precision point tracking errors, $e_{1\rightarrow3}$. Shown as well are the maximum tracking error for each mechanism and the sum of the mechanism's tracking errors.

Using the fitness function of Equation 4.3, solutions Z and W receive the same fitness score. Z and W, however, attempt to solve the problem in

Individual	e_1	e_2	e_3	$Max(E)$	Σe_i
Z	0.33	0.33	0.33	0.33	0.99
W	0.99	0.0	0.0	0.99	0.99

Table 4.3: Different Solutions, Same Σe_i. Three precision points, two candidate solutions.

vastly different manners; Z attempts to minimize the tracking error at each prescribed point while W goes perfectly through two points and takes all of the performance 'hit' on one prescribed point. These two solutions represent two competing strategies, and hence two competing genetic makeups, for solving the path error minimization problem. As the GA cannot distinguish the better genetic makeup, convergence problems arise. As in Roston and Sturges (1996) preliminary results in the current study showed that this fitness relationship tends to promote convergence to local maxima.

A relationship having better global search performance is presented by Roston and Sturges (1996) for minimizing the sum of the tracking errors - theirs was a dynamic fitness function that tightened error tolerance with better solutions in the population. Kim (1995) similarly based his fitness function on minimizing the sum of the path tracking errors. In the current study we will consider minimizing not the sum, but the maximum tracking error in the set, as this is the criteria on which the solution will later be judged (*i.e:* "The mechanism passes through the prescribed locations with a maximum tracking error of 0.025."). Our fitness function formulation:

$$Fitness = \frac{1}{1 + Max(E)} \tag{4.4}$$

not only relieves any ambiguity as to the quantity being minimized, it clearly represents our design goal to find the mechanism with the smallest path tracking error.

Figure 4.9 plots this relationship between the maximum path tracking

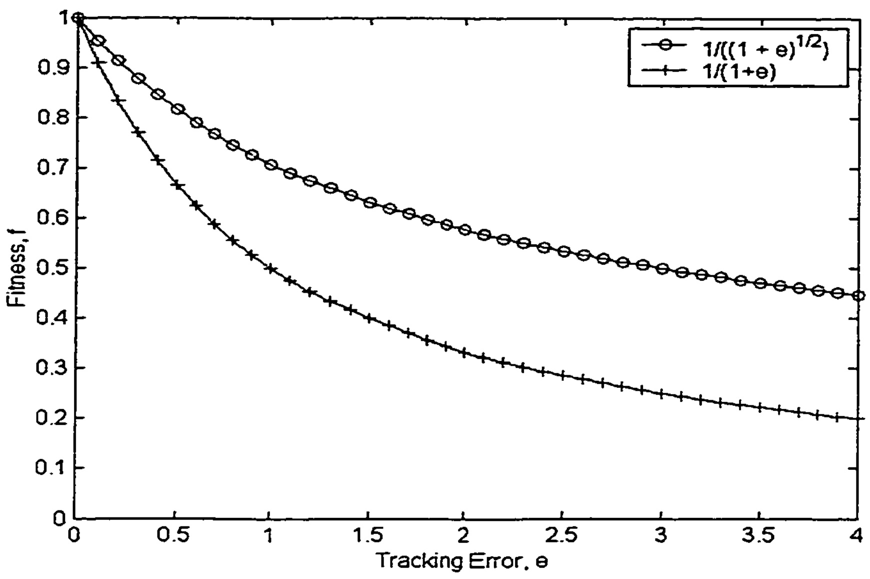

Figure 4.9: The fitness function: transforming $Max(E)$ into a fitness value.

error and assigned fitness. A variation of Equation 4.4,

$$Fitness = \frac{1}{\sqrt{1 + Max(E)}}, \tag{4.5}$$

having a lesser rate of decay for most of the error range is shown as well. It is theorized that the more gradual transition from low to high fitness scores of Equation 4.5 promotes exploration of a greater number of sub-optimal regions as members of these regions garner higher fitness scores. This wider search is expected to have a better possibility of finding the global optimal solution within the search region.

In order to compare the searching ability of the two fitness functions, the 16 prescribed point experiment will be run repeatedly with each function. The GA inputs and search variable ranges of Section 3.3.1, together with the optimal population size found in that section, will be used. The GA will be run 50 times using each fitness function formulation; the number of times out of 50 that the algorithm converges on the known solution or a similarly high quality solution will be interpreted as the probability of finding an excellent solution in one run of the GA with that fitness formulation.

4.4.3 Problem design complexity tests

Having specified the GA parameters needed for successful searches for a given problem and search landscape size, we wish to investigate the effect of the complexity of the design task on search performance. That is, how does the number and arrangement of specified points affect the probability of the algorithm finding the known solution. As greater numbers of specified points place more constraints on the design task, it is reasonable to assume that the number of potential solutions contained within the search space is reduced. We wish to observe the effect this has on the probability of finding the known solution in a single run of the GA.

To test this, we perform design tasks on two different arrangements each of point sets having 4, 8 and 16 prescribed points. The point sets are generated from the known mechanism geometry of Section 3.3.1. Both a closely spaced point scheme and a widely spaced point scheme are selected arbitrarily for each of the 4, 8 and 16 prescribed point tests, for a total of 6 different prescribed point arrangements. The two different point spacing schemes for each number of precision points are considered in order to gauge the effect of point spacing as well as number on algorithm search ability. The prescribed point sets are shown in Figure 4.10. Using the best input parameter set of Section 3.3.1 and the best objective function of Section 3.3.2, the GA is run until completion 50 times on each point set. The number of times the algorithm finds an excellent solution to the problem will be used to approximate the probability of the algorithm finding an excellent solution in a single run.

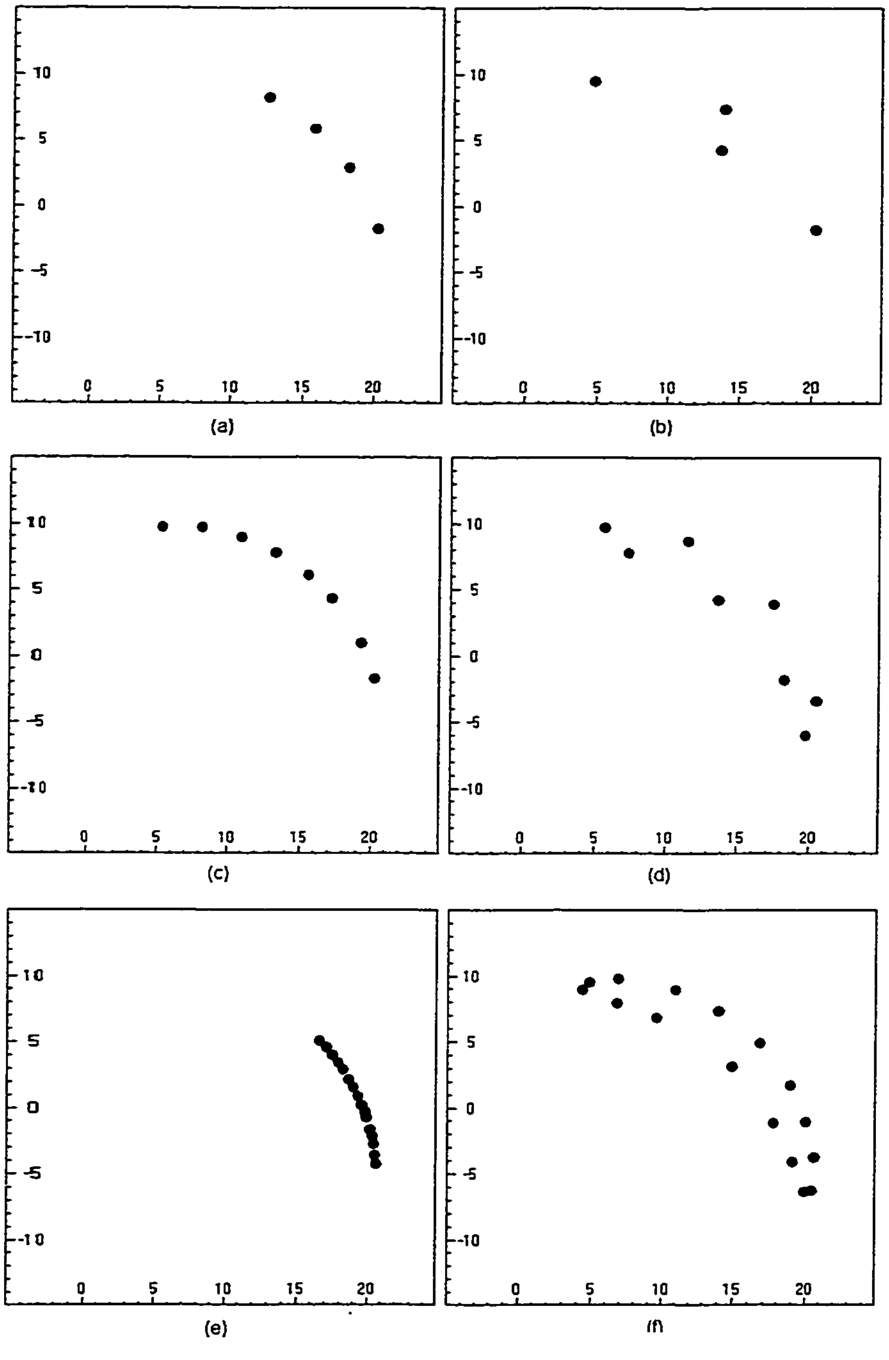

Figure 4.10: The experimental point sets. Schemes a, c and e are closely spaced points, b, d and f are relatively evenly spaced points.

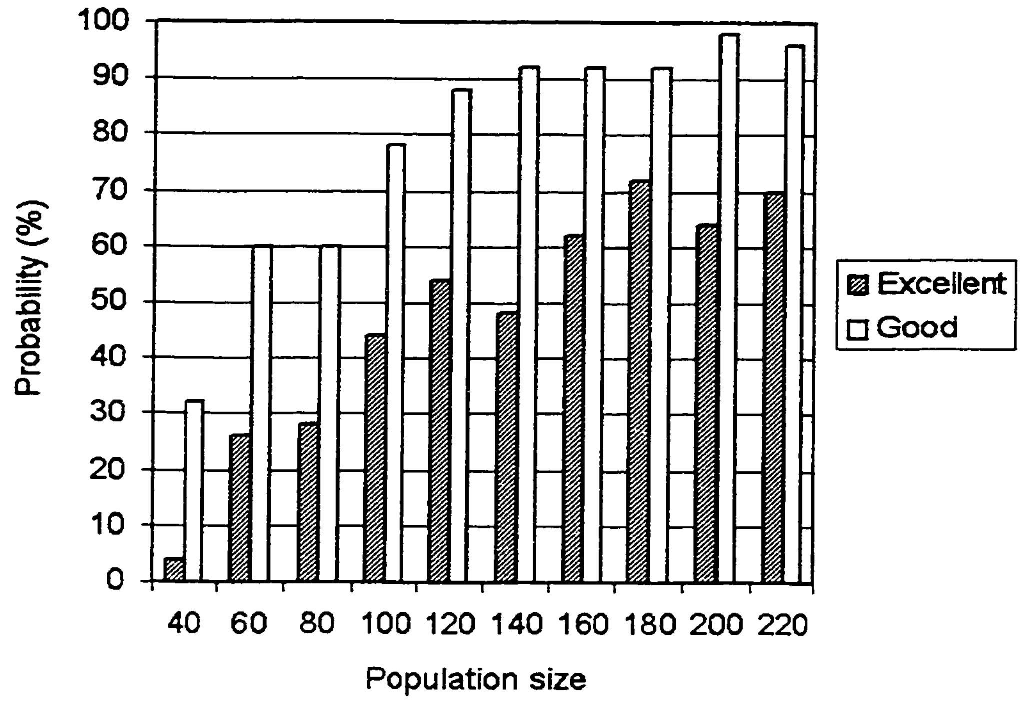

Figure 4.11: Probability of finding good and excellent solutions in one run of the GA for increasing population size.

4.5 Results

4.5.1 Appropriate GA inputs

Figure 4.11 shows the results from running the GA 50 times on each population size for the problem shown in Figure 4.6, having 16 specified points. As expected, the probability of finding both good and excellent solutions in a single run of the GA on the 16 point problem increases with population size. Population sizes less than 120 have a less than 50% chance of finding an excellent solution in a single run of the GA. Those having less than 100 individuals can be expected to find a good solution in a single run only 60% of the time and an excellent solution only 30% of the time.

The results of Figure 4.11 emphasize two main points that must be considered when searching with the GA:

1. Finding globally optimal or near-optimal solutions is not a certainty with a single run of the GA. Even with optimal input parameters the GA will occasionally not find the global maximum, hence several runs to completion should be performed and the best results kept.
2. Small population sizes yield many near-optimal solutions. In order to find the global optimal solution(s) in the search space, large population sizes are often required.

With larger population sizes comes not only increased success probability, but also many more function evaluations.

Figure 4.12 shows the average number of generations required for the GA to converge on a solution for each population size. Not unexpectedly, larger population sizes require more generations to converge as more individuals have to be coerced into resembling the best found solution. The number of potential fitness function evaluations in the course of a GA run increases substantially with population size as well; not only does the GA require more generations to converge with larger populations, but the larger population sizes equate to more function evaluations per generations as well.

The question arises as to what is the best population size strategy. Should the GA be run several times until completion with a small population size, or just a few times using a very large population size? In order to compare the probability of finding the known optimal solution with different population sizes and numbers of GA runs, we will define the probability of success, $P_{Success}$, as the probability of finding the global optimal solution at least once in several runs of the GA for a given population size. Knowing the probability of finding an excellent solution, $P_{Excellent}$, in a single run for a given population size, we can estimate the probability of success as

$$P_{Success} = 1 - (1 - P_{Excellent})^k \tag{4.6}$$

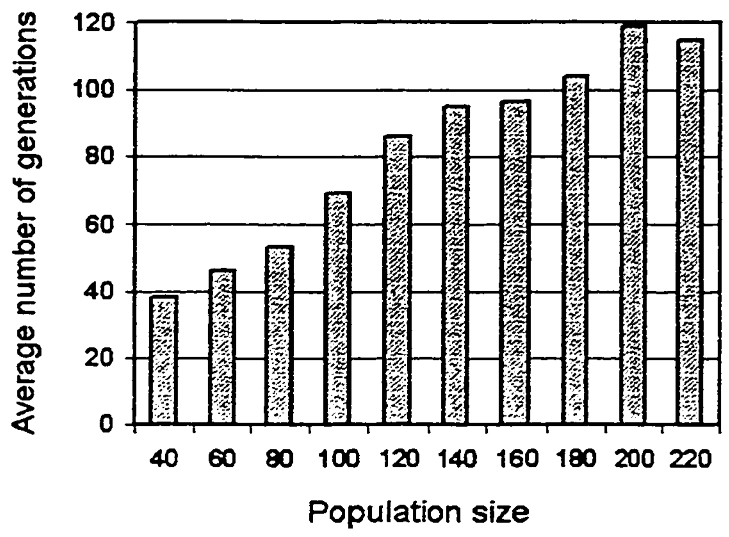

Figure 4.12: Average number of generations before GA convergence as population size is increased.

where k is the number of experiments, or runs, performed. Hence if we run the GA twice with a population size of 160 (where $P_{Excellent} = 62\%$), we will have a probability of

$$P_{Success} = 1 - (1 - 0.62)^2 = 0.856,$$

or 85.6%, of finding an excellent solution at least once in the 2 runs.

In order to compare the searching ability of the different population sizes, we need to calculate the number of runs, and hence function evaluations, required to attain a desired success probability. Specifying $P_{Success} = 99\%$, the number of runs for each population size to attain this level can be calculated using Equation 4.6 with the probability of finding an excellent solution for that run (Figure 4.11). For example, knowing that $P_{Excellent} = 26\%$ for a population size of 60, the number of runs required to achieve a 99% chance of finding an excellent solution is predicted by:

$$0.99 = 1 - (1 - 0.26)^k.$$

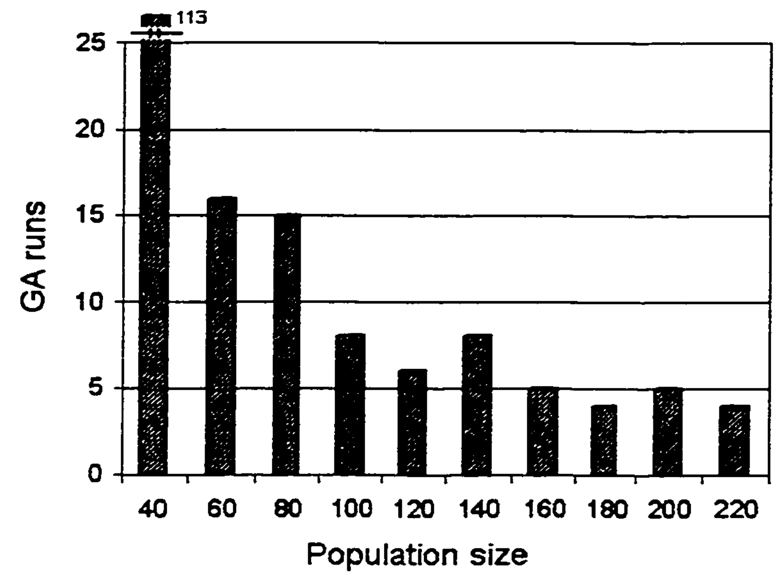

Figure 4.13: Number of GA runs required for each population size to achieve $P_{Success} > 99\%$.

Solving, we find that $k = 15.29$, hence 16 runs of the GA with a population size of 60 are required to have a 99% chance of finding the global optimal solution at least once. Figure 4.13 shows the number of runs required for each population size.

Knowing the number of runs required, we can estimate the number of function evaluations needed to attain $P_{Success} = 99\%$ by multiplying the number of runs by the average generations per run and the number of individuals in the population. This is an upper limit[2] on the number of function evaluations required for each population size as we are assuming that n function evaluations occur for each generation of a population of size n. Figure 4.14 shows this upper limit on the number of function evaluations required to obtain our desired $P_{Success}$ for each population size.

Figure 4.14 indicates that the strategy requiring the fewest function eval-

[2]In practice, problems having more lengthy function evaluations would store each evaluated solution in memory, calculating all explored input variable combinations just once.

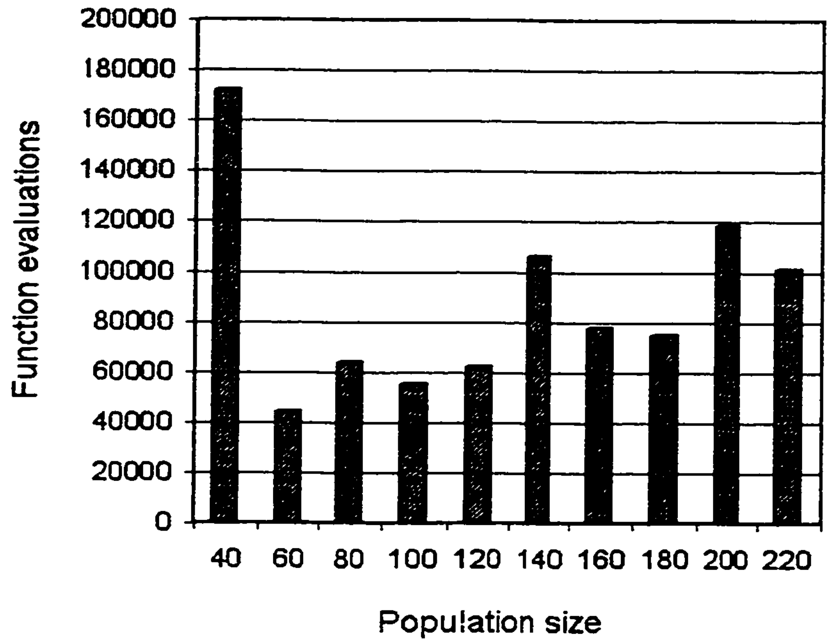

Figure 4.14: Number of function evaluations required for each population size to achieve $P_{Success} > 99\%$.

uations (44,160) is to run a population size of 60 individuals 16 times (Figure 4.13) and select the best individual from the best of each run. The population size of 100 was the second best strategy, requiring 55,200 function evaluations, which corresponds to 8 runs of the GA. The larger population sizes required only 4 to 5 runs to attain this performance level, yet due to their large sizes required more function evaluations and hence longer search times. It seems that a computationally efficient approach is to run a smaller population size many times and select the best individual from the best solution of each run. The population sizes predicted by Goldberg (1989a) are far larger; this is because his analysis was looking for an optimal population size for finding global optimums within a single run while the present analysis considers that there is the possibility of doing several runs.

Index	Location
1	(18.12276, 3.339924)
2	(14.31492, 7.309055)
3	(13.01194, 4.628347)

Table 4.4: Three prescribed point locations chosen from the set generated using a crank link driving increment of 0.5°.

Item	Closest Discrete (L_{cd})	Circle Approximation (L_c)
e_1	7.312×10^{-2}	1.500×10^{-7}
e_2	6.749×10^{-2}	2.995×10^{-6}
e_3	2.368×10^{-1}	3.905×10^{-6}
Fitness	1.899	2.000

Table 4.5: Tracking errors ($e_{1\rightarrow 3}$) assigned to each prescribed point and solution fitness based on L_{cd} (distance to closest discrete path point) and L_c (circle fit approximation) measures.

4.5.2 Objective function formulation

Path tracking error calculation

The coordinates of the three points chosen from the discrete path point set generated by driving the crank link in 0.5^o increments are shown in Table 4.4. Superimposing these three points on the set formed by using a driving crank increment of 1^o yields Figure 4.15(a). An exploded view of the square region in the figure, including two of the three points, is shown in Figure 4.15(b).

We can now consider the effect of defining our path tracking error for each prescribed point as the distance to the closest discrete path point, L_{cd}, as well as the distance L_c as predicted by Kim's circle fit approximation to the curve. Table 4.5 shows the tracking error magnitude for each of the three chosen prescribed points using both the L_{cd} and L_c measures. Equation 4.5 Shown as well are the fitness scores assigned to the mechanism (using Equation 4.5) having the known geometry based on these three points.

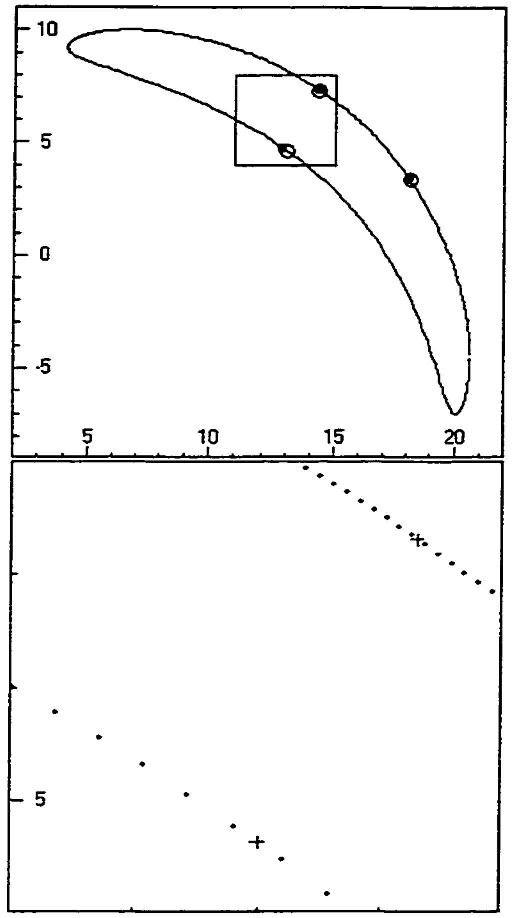

Figure 4.15: (a): Locations of three points (hollow circles) taken from the coupler curve generated with a driving crank increment of 0.5^o. (b): An exploded view of the square region of (a). Small circles are discrete points generated using a crank driving increment of 1^o, crosses represent prescribed points 2 and 3 taken from the path of the same mechanism using a 0.5^o increment.

Fitness Function	$P_{excellent}$	P_{good}	Average Generations
Equation 4.4	33	66	116
Equation 4.5	37	70	99

Table 4.6: Tracking error to fitness relationships.

Knowing that both the prescribed point set and the path point set were produced by identical mechanisms but with different driving crank increments, we would hope that the tracking errors assigned to each prescribed point for each method would be negligible. As Table 4.5 shows, the scores calculated using the circle fit approximation correctly assign negligible tracking errors to the prescribed points, while the scores calculated using the closest discrete point distances do not.

The largest tracking error in the set ($e_3 = 0.2368$) calculated using the discrete point distances is far greater than the limit required for a good solution ($2\delta = 0.0628$). Hence when using the closest discrete path point approximation, even those mechanisms having the exact geometry may be erroneously assigned large path tracking errors. Kim's circle fit approximation, however, predicts accurate path tracking errors for the chosen driving crank increment (1^o).

Tracking error to fitness score function

Table 4.6 shows the results of running the GA 100 times on our 16 prescribed point problem using both the tracking error to fitness relations of Equations 4.4 and 4.5. Equation 4.5 shows slightly better performance, having a higher probability of finding both excellent and good solutions in a single run. Additionally, the average number of generations required for the runs using this function is lower than for the one given by Equation 4.4.

Scheme	$P_{excellent}$	P_{good}	Average Generations
4A	74	94	42
4B	38	78	37
8A	62	92	65
8B	42	94	42
16A	60	92	33
16B	54	88	86

Table 4.7: Problem complexity results.

4.5.3 Design complexity tests

Results of the design complexity tests are shown in Table 4.7, where, in the first column the numeral represents the number of specified points and the letter is the spacing scheme (A = closely spaced, B = widely spaced). For each experiment the probability of finding good and excellent solutions in a single run of the GA is shown along with the average number of generations required for GA convergence.

The point spacing scheme results show great variation in the probability of finding good and excellent solutions. Considering the closely spaced (A) schemes, it becomes less probable to find an excellent solution as the number of prescribed points is increased. The B (widely spaced) schemes show the opposite trend: it is more likely that an excellent solution will be found when more points are prescribed. Clearly there is an interdependence between the number and spacing of the prescribed points and the probability of finding an excellent solution in a single run of the GA.

For a given number of specified points, the closely spaced scheme experiments were more likely to find an excellent solution. The average number of generations required for each experiment to converge on a solution shows little variation, except for the 16B and 8A experiments. These required several more generations to converge than the others. This could be due to the presence of several near-optimal solutions for these prescribed point sets. The 16A experiment appears to be the opposite, requiring few generations, on average, to converge on a solution. This is most likely due to the fact that the 16A point set is approximated by a small portion of the solution space, hence there are few good solutions to choose between.

4.6 Conclusions

This chapter has shown some of the merits and downfalls of using the GA as a design tool for this class of problem. The main aspects considered while using the GA to design four bar linkages for path generation tasks, were:

- finding appropriate GA input values,
- formulating an objective function, and
- design complexity considerations.

The GA appears to be a feasible design tool for this class of problem, provided that a proper population size and number of GA runs is used, an objective function is formulated that is meaningful to the design goals and the designer is sensitive to design complexity considerations.

In all of the experiments it is apparent that several runs until completion of the GA are required to find global optimal solutions. In particular, the population size parameter has a significant impact on the probability of the algorithm finding global optimal or near optimal solutions. A single run of the GA with a reasonably large population has a good chance of finding near optimal solutions. To further increase the probability of finding global optimal solutions, the algorithm should be run several times until completion. A computationally efficient method appears to be to use a smaller population size and several runs of the algorithm until completion.

Formulating an objective function is certainly one of the most critical aspects of using the GA in such a design task. This is truly a case of 'what goes in, comes out' as an objective function definition not representative of the desired design will typically yield undesired solutions. The GA's ability to find solutions which satisfy the objective often yields unexpected solutions to the design task. In the present study the addition of the circle fit approximation when calculating the path tracking error was significant: without the approximation it was shown that those paths running through the prescribed points could incorrectly be assigned non-negligible error values.

The complexity and 'shape' of the design space is a consideration for likely all numerical optimization methods. It is prudent for the designer to make themselves aware of the variations in the search landscape. Fortunately, if the GA is run several times on a problem, the frequency and locations of maxima found can offer some insight as to what can be expected of the landscape. How large a portion of the design space good solutions occupy is a constant concern for nearly all optimization methods; finding a needle in a hay stack is a hard job for any search technique. The algorithm performance is therefore bounded in a manner by the nature of the solution space.

This chapter provides important insight into applying the GA to a typical design task. In the following chapter these findings will be applied to aid in using the GA for other design tasks.

Chapter 5

Mechanical prosthesis design using the GA

5.1 Chapter Glossary

- L_1 - Length of the stance leg (m).
- L_2 - Length of the swing leg thigh (m).
- L_3 - Length of the swing leg shank (m).
- m_i - Mass of the ith segment (kg).
- I_i - Mass moment of inertia of the ith segment about its center of mass.
- β - Knee joint flexion angle (degrees).
- K_{assist} - Knee joint assist spring coefficient (Nm/radian).
- C_{assist} - Knee joint assist damper coefficient (Nms/radian).
- K_{lock} - Knee joint locker spring coefficient (Nm/radian).
- C_{lock} - Knee joint locker damper coefficient (Nms/radian).
- β_{lock} - Knee joint lock active range (degrees).

- β_{damper} - Knee joint assist damper active range (degrees).
- M_{hip} - Applied hip flexor moment (Nm).

5.2 Introduction and Background

Having gained a better understanding of what parameter values are required for successful search using the GA, the first of two 'real-world' design problems will be considered. This chapter will focus on designing a purely mechanical above the knee prosthetic leg. The goal is to find the parameters for a chosen mechanism type which result in the most natural looking swing phase of human gait when using the prosthesis. The mechanism design problem reduces to a dimensional synthesis task for the chosen mechanism.

The goal is to explore the feasibility of designing a purely mechanical prosthetic knee joint that exhibits natural looking gait. Current prosthetic limbs available for people with mid-thigh amputations range from the basic (simple wooden legs) to the exotic (dedicated microprocessors controlling mini-machinery). Costs for these prostheses range similarly, starting low and rapidly increasing.

The purpose of this experiment is to illustrate using the GA as a design method. This particular application seems suitable to explore the power of the GA as it will be shown that the solution space is very complex and the best choice of function to be optimized is somewhat uncertain.

Similar optimization applications can be found in the gait analysis literature (Davy and Audu, 1987) as researchers have used advanced search techniques to predict such things as the optimal control signals sent to muscles in the lower limb during human gait. The current study is a departure from the published works in that a global search technique, the GA, is used as the design method. It will be shown that the GA finds optimal parameters for the simple knee joint model; this is justification for using the method to optimize more complex designs involving more design variables.

A simple dynamic model of human gait in the swing phase of locomotion is used. The knee joint will be mechanically controlled by spring and damper elements. It is the properties of these elements that will be sought, using the GA to yield a design having natural looking swing.

5.3 Method

Two models for the knee joint design experiment will be combined: a gait model that simulates the dynamics of a real person walking and a knee joint model that is the mechanism to be designed. The knee joint model will affect the motion of the gait model, hence it is the best parameter set of the knee joint model that we seek to yield the most natural looking swing in the gait model.

5.3.1 The gait model

Our gait model will be a planar three degree of freedom (3DOF) model considering only the swing phase of human gait. Swing phase is defined as the single support period of human gait, or the period from when one foot leaves the ground to subsequent touch-down, or heel-strike. We will use a model similar to that used by Mochon and McMahon (1980), composed of three rigid links connected by revolute joints to a fixed ground. Figure 5.1 shows the stance leg, L_1, the swing thigh, L_2, and the swing shank, or lower leg, L_3. The system orientation for any time is defined by three independent generalized coordinates ϕ_1, ϕ_2 and ϕ_3 which are the angles of each link measured with respect to a fixed vertical reference frame. The angle β is the knee flexion angle of the swing leg - this will be of primary concern in our analysis.

Data for the segment properties have been taken from Gaitlab (Vaughn *et al.*, 1992). Gaitlab presents data for a healthy male subject; the file 'Man.seg' fully specifies the length, mass, mass moment of inertia and center of mass for each segment of our model. The values reported for the shank

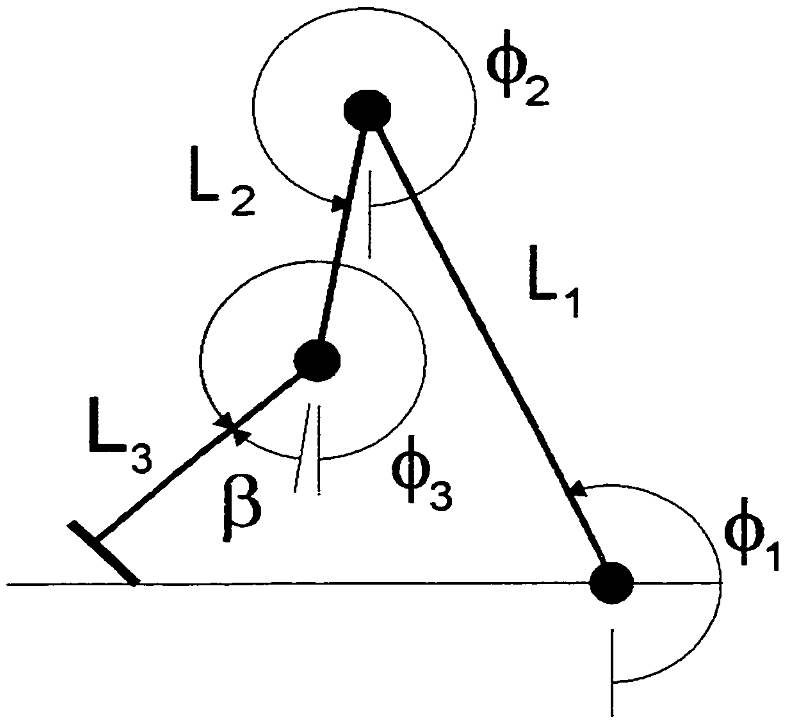

Figure 5.1: Three bar ballistic walker. Link L_1 is the stance leg, L_2 is the swing thigh and L_3 is the swing shank.

Variable	Gaitlab	Model
ϕ_1	3.563	3.563
ϕ_2	6.11	6.11
ϕ_3	5.47	5.47
$\dot{\phi}_1$	-1.65	-2
$\dot{\phi}_2$	1.94	1.94
$\dot{\phi}_3$	-1.72	-1.72

Table 5.1: Initial conditions as reported in Gaitlab and as used.

and foot of the male subject were combined to yield a composite segment representing the lower limb of our model.

The model is a ballistic one, in that the only externally applied force on the system is due to gravity. The equations of motion of this 3DOF planar system are:

$$[I]\{\ddot{\phi}\} + [M]\{\dot{\phi}^2\} + \{G\} = \{0\}, \tag{5.1}$$

where the coefficients of $[I], [M]$ and $\{G\}$ are presented in Appendix C. Given a set of initial conditions on $\{\phi\}$ and $\{\dot{\phi}\}$, Equation 5.1 can be integrated numerically to find the subsequent position, velocity and acceleration of each body in the system from time $t = t_{toe-off}$ to time $t = t_{heel-strike}$.

Values for both the initial conditions and the time of swing were based on Gaitlab data for the healthy male subject. Table 5.1 presents the initial conditions used in the simulation and those presented by Gaitlab. The velocity of the stance leg of our model had to be increased slightly to ensure that the model did not fall backwards, otherwise the initial conditions are identical to the real data presented for the male subject. The time of swing, or the time that is spent in the single support phase of gait, was held fixed for all of our simulations at $t_{swing} = 0.5$ seconds, or a slow walking speed. The performance of the model at other walking speeds was not considered.

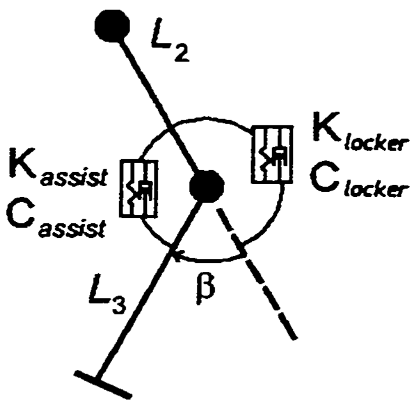

Figure 5.2: Knee joint connecting links 2 and 3. β is the knee flexion angle, k and c are the spring and damper constants.

5.3.2 The knee joint model

The swing leg knee joint will be altered as shown in Figure 5.2. We have added to the revolute joint two 'assist' elements: a rotational spring and a rotational damper. The rotational spring is unstretched when the knee is fully extended ($\beta = 0$ degrees) and opposes knee flexion ($\beta > 0$ degrees), forcing the knee joint to extend. The damper opposes the spring reaction in the knee flexion range $\beta > \beta_{damper}$. This element is active in one direction only, when $\dot{\beta} < 0$ degrees/second, or when the knee is extending. A bumper, needed to prevent hyperextension ($\beta < 0$) of the model's knee joint, is represented as a spring and damper combination, having parameter values of K_{locker} and C_{locker}, that is active when $\beta < \beta_{locker}$. These four elements make up the knee joint model. The motion of the gait model through swing phase is influenced by the knee joint model, hence it is the properties of these elements that we will search to find the mechanism having the most natural looking gait.

We will consider three variables specifying the assist spring constant, K_{assist}, the assist damper constant, C_{assist}, and the assist damper cutoff value, β_{damper}, to be varied to find the mechanism having the most natural looking gait. We will hold the locking mechanism properties constant, having values of K_{locker} = 140 Nm/radian, C_{locker} = 8 Nms/radian and β_{locker} = 3 degrees, determined by simple experimentation to give a natural locking appearance. As detailed in the following section, we will base our assessment of solution quality on the difference between a solution's knee flexion curve, the plot of the swing leg knee flexion angle with respect to time throughout swing phase, and that of a reference knee flexion curve.

To test our search method and determine optimal GA search parameters, we will first search for a known mechanism in the solution space. The knee flexion curve of our known mechanism, having parameter values K_{assist} = -1 Nm/radian, C_{assist} = 1 Nms/radian and β_{assist} = 13 degrees, as well as that for our gait model having no assist elements, is shown in Figure 5.3. It is evident that the knee model changes the dynamics of the gait model considerably, based on the knee element properties.

Our initial experiments will therefore search for the set of K_{assist}, C_{assist} and β_{damper} that yield a knee flexion curve most closely resembling that of our known mechanism. By varying the GA input parameters such as the population size, we can determine appropriate inputs required for the GA to successfully search this size of a solution space. These inputs will then be used to search for the mechanism parameters yielding a knee flexion curve most closely resembling that of real human gait.

5.3.3 Setting up the GA

When using the GA, finding both a proper objective function to minimize or maximize and a set of GA input parameters that will result in successful search performance is essential. The objective function must relate the desirable qualities of a good problem solution to a quantity that reflects how good a particular solution is. Therefore an objective function must yield high scores for those solutions that exhibit most of the desirable qualities and small scores for those that do not.

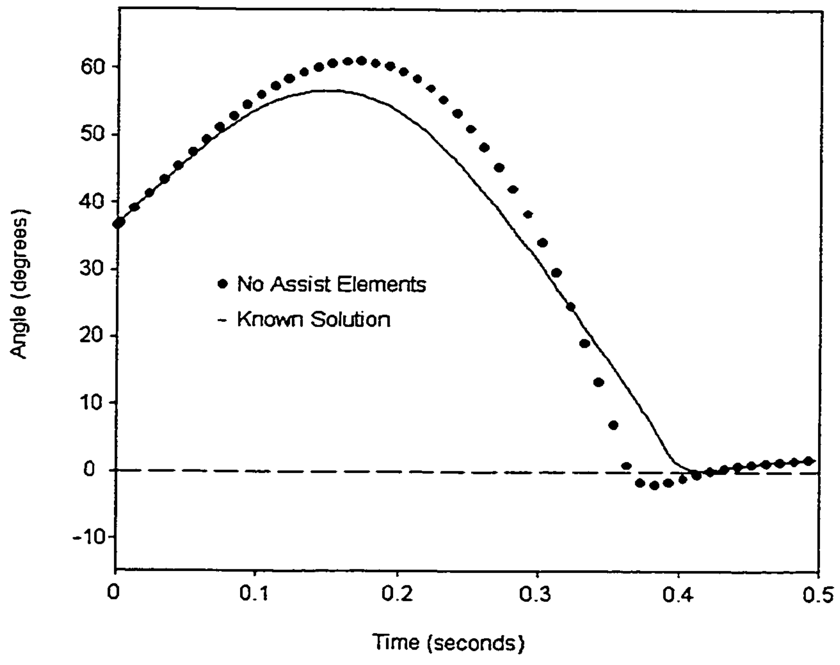

Figure 5.3: Knee flexion curves for the known solution having K_{assist} = -1 Nm/radian, C_{assist} = 1 Nms/radian and β_{assist} = 13 degrees and the gait model having only a locker mechanism and no assist elements.

We will define our indicator of natural looking gait based on the similarity of the model's knee flexion curve, that of β with respect to time, to that of the real human data that our gait model mass quantities are taken from. The objective function will be based on the root mean square (RMS) difference between the reference knee flexion curve and that of the current solution at each time step. The reference and tested solution will be compared in the time interval from 0 to 0.5 seconds at 0.002 second increments. This time increment is the numerical integration step size.

The RMS tracking error, e_{RMS}, for a particular solution will then be turned into a fitness score using one of two methods. The first is to simply take the reciprocal of the RMS tracking error as the fitness score, hence

$$Fitness = 1/e_{RMS}. \tag{5.2}$$

An alternate method is to use the tracking error to fitness formulation of the previous chapter, ie:

$$Fitness = 1/\sqrt{1 + e_{RMS}}. \tag{5.3}$$

Both methods will be examined to see if there is any difference in algorithm performance.

The population size of the GA is dependent on the size of the space being searched. A too small population size will often lead to convergence to suboptimal local maxima. Population sizes that are too large will lead to excessive generations required for convergence. The crossover and mutation rates exhibit similar traits - there are optimal settings for each. As in the previous chapter, a medium crossover rate (65%) and a mid to high mutation rate (0.8%) will be used as it was shown that this is a good strategy. We will rely on the problem independent nature of the GA by finding a population size that consistently finds the known solution contained within the solution space to then search for the parameters that yield the best solution on the real problem.

The known solution to search for has parameters as shown in Table 5.2. The negative sign convention on the spring stiffness parameter is due to the

Parameter	Value
K_{assist}	-1 Nm/radian
C_{assist}	1 Nms/radian
β_{damper}	13 degrees

Table 5.2: The known leg joint solution parameters.

Parameter	Lower Limit	Upper Limit
K_{assist}	0 Nm/radian	-5 Nm/radian
C_{assist}	0 Nms/radian	5 Nms/radian
β_{damper}	0 degrees	20 degrees

Table 5.3: Search variable ranges.

sign convention for the knee joint angle used. This means that for some knee angle flexion, the spring is in compression, generating a force which tries to extend the lower limb. The knee flexion graph for this mechanism is shown in Figure 5.3. Limits on the search variables are shown in Table 5.3. Knee flexion graphs of two mechanisms having extreme values of the search variables are shown in Figure 5.4, giving a crude indicator of the possible mechanism knee flexion curves contained within the solution space being searched.

In order to compare the quality of solutions found in a run of the GA, a high quality solution needs to be specified in the context of this problem. As the quantity to be minimized is the RMS tracking error between the two curves, this value will be the basis of the comparison. The RMS tracking error between the two curves, e_{RMS}, will be compared to the finest input variable increment, δ, in the β_{damper} variable. Those solutions having $e_{RMS} \leq \delta$ will be considered 'excellent'. Shown in Table 5.4 are the RMS tracking errors for the known solution, the closest solution representable by the GA, and two variants of the latter; one having each parameter value one increment larger than the closest, the other having each parameter one increment less.

The population size experiments will be performed by running the GA 20

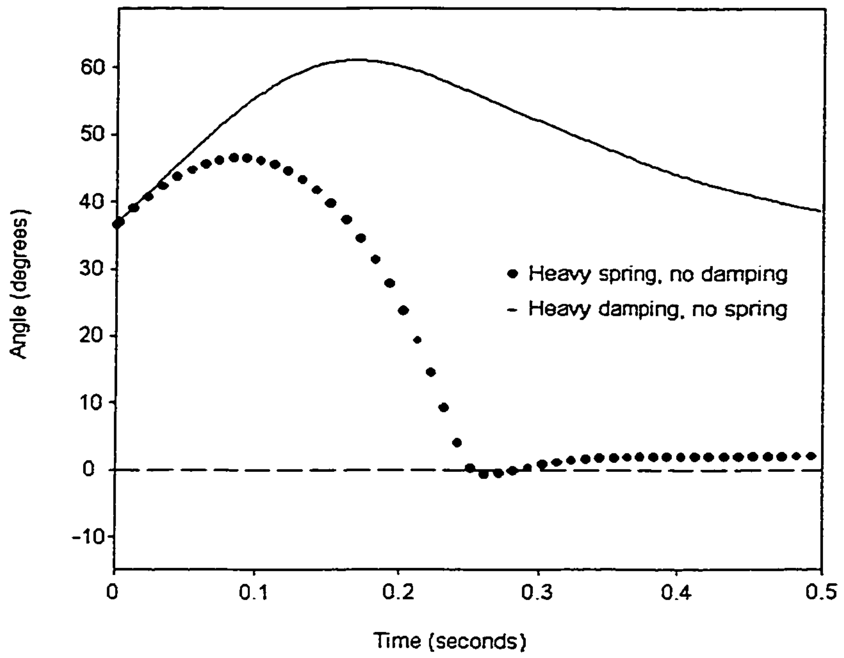

Figure 5.4: Knee flexion curve extents. Shown are curves for mechanisms having a heavy spring and no damping (K_{assist} = -5 Nm/radian, C_{assist} = 0 Nms/radian) and heavy damping and no spring (K_{assist} = 0 Nm/radian, C_{assist} = 5 Nms/radian and β_{assist} = 20 degrees).

Parameter	(a) Known Value	(b) Closest GA	(c) Variant1	(d) Variant2
K_{assist} (Nm/radian)	-1	-1	-0.9804	-1.0196
C_{assist} (Nms/radian)	1	1	1.0196	0.9804
β_{damper} (degrees)	13	13.0196	13.0980	12.9412
e_{RMS} (degrees)	0.0	0.0	0.0798	0.0740

Table 5.4: Solution parameters and RMS tracking errors for (a) the known solution, (b) the closest discrete GA representation and (c) and (d), two variants with parameters close to those of (b).

times each with populations ranging from 40 to 160 individuals in steps of 20. The number of runs that the GA finds the known solution, or a similar high quality variant, will be interpreted as the probability of success for a single run with that population size. The population size with the highest probability of finding the known solution will then be used to search for the parameter set yielding a knee flexion curve most resembling that of human gait.

5.3.4 Searching for the parameter set yielding natural looking gait

Having determined an optimal population size and the most suitable fitness function in the previous experiment, the parameter set resulting in a knee flexion angle most similar to that of natural gait will be sought. Figure 5.5 shows the knee flexion angle graph taken from the 'Man.ang' file found in Gaitlab. This data represents the natural knee flexion curve for the gait of a normal man upon which the three link model mass and geometric properties are based.

The Gaitlab data file presents, among many other things, knee flexion data for the entire gait cycle. As we are concerned only with swing phase of gait, we have taken data points starting from the point of toe-off, t_{off}. Toe-off, identified as the point at which the force plate data showed a zero load for the swing leg, occurred at $t_{off} = 0.82s$. The time of heel strike was undefined for the data set as only one force plate was used per foot. The period of swing phase was defined by our model; a period of $t_{swing} = 0.40s$ typically was needed before heelstrike occurred. This is typical of reported swing period times (Mochon and McMahon, 1980). It was decided to use Gaitlab data for a period of 0.5 seconds in order to include the knee-locking behaviour evident in the 0.1 seconds of the knee angle plot. This is perhaps not a good assumption as the knee angle behaviour in this time period is likely due in part to heel strike occurring. Regardless of the assumption validity, it does make for a more difficult target to attain, so the data taken

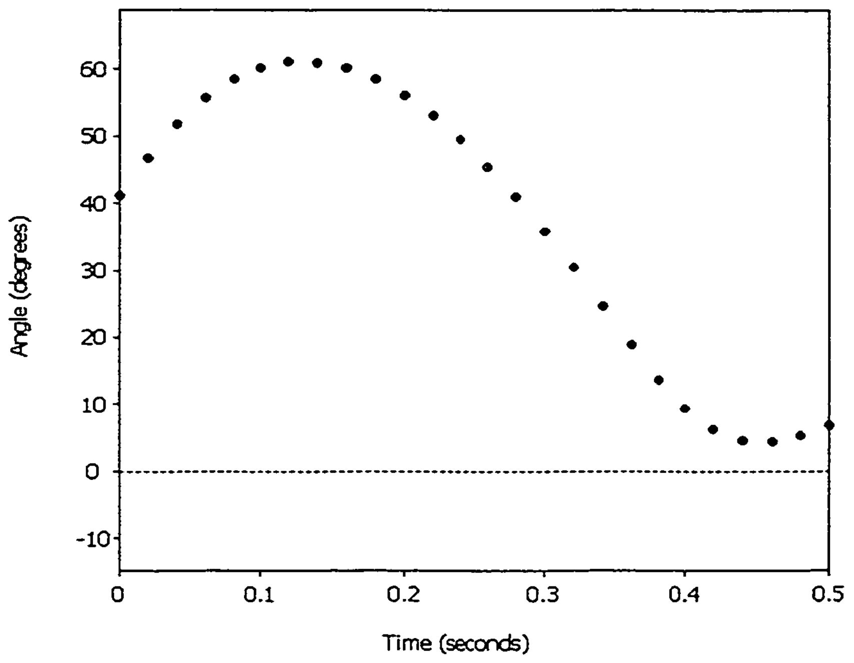

Figure 5.5: Knee flexion angle data for swing phase of gait of a healthy male subject. Taken from Gaitlab.

from Gaitlab will be for a total of 0.5 seconds. The reference data set is therefore made up of the knee flexion angle data reported in the Gaitlab data file from time $t = 0.82s$ to time $t = 1.32s$ in 0.02 second time steps.

5.4 Results

5.4.1 GA fitness function and population size required

GA fitness functions

It was found that the form of the fitness function greatly affected the algorithm performance with respect to the algorithm converging on a solution. Interpreting fitness as simply the reciprocal of the RMS tracking error produced populations of solutions where the difference between the best and worst individuals is dramatic. This resulted in unsatisfactory algorithm performance; the algorithm often failed to converge on a solution, hence Equation 5.2 was deemed unacceptable. Equation 5.3, having an upper bound on the good individuals' fitness scores was therefore determined to be the best strategy.

It was also found that if the quantity to be minimized was not the RMS tracking error, but the largest absolute tracking error in the set, the GA did not find acceptable solutions. When attempting to minimize the largest absolute tracking error, those solutions having many small deviations from the reference curve would be awarded high fitness, while those following the general trend of the reference curve, yet deviating significantly in one small region were penalized heavily. As the goal is to find a joint configuration that looks most natural, it makes sense that the solution should have the same general pattern as that of natural gait.

Population size required

Table 5.5 shows the results of the population size tests. Shown are the number of times, of a total 20 runs per population size, that the algorithm found the

Population Size	40	60	80	100	120	140	160
# Success	5	11	11	16	15	17	19

Table 5.5: Population size test results. Number of runs, of a maximum 20 that found a best solution ever having an 'excellent' tracking error score.

equivalent of the known solution, or an 'excellent' solution to our problem. As expected, there seems to exist a minimum population size required for successful searches. Population sizes of 100 or greater were found to yield the best results for this size of search space. This was not surprising as the search space for the present problem is the same size as for the problem of the previous chapter ($2^{Parameters \times Bits/Parameter} = 2^{3\times 8} = 2^{24}$) where it was found that a population size of 100 or greater was required. Using a population size less than 100 individuals often resulted in the algorithm converging on non-optimal, or local maxima, solutions.

5.4.2 Search results for natural looking joint design

Figure 5.6 shows the best solution found after running the GA 10 times on the real joint design problem. This joint, having parameters K_{assist} = -0.6471 Nm/radian, C_{assist} = 1.2549 Nms/radian, β_{damper} = 0.825 degrees resulted in an RMS tracking error of 18.564 degrees. Again, the negative sign on the spring rate is a result of the knee flexion angle sign convention. While the general trend of the best solution is similar to that of natural gait, our model does not attain the same peak knee flexion as that of real gait and the region around heel strike ($t \geq 0.45s$) differs significantly.

As is often the case in GA searches, other high quality solutions having different parameters values were found. Table 5.6 lists a few of the high quality variants found, showing the solution parameters and resulting RMS tracking error for each.

The difference in peak knee flexion is reasonable as our model is purely ballistic, assuming that no muscles act throughout swing phase. In real-

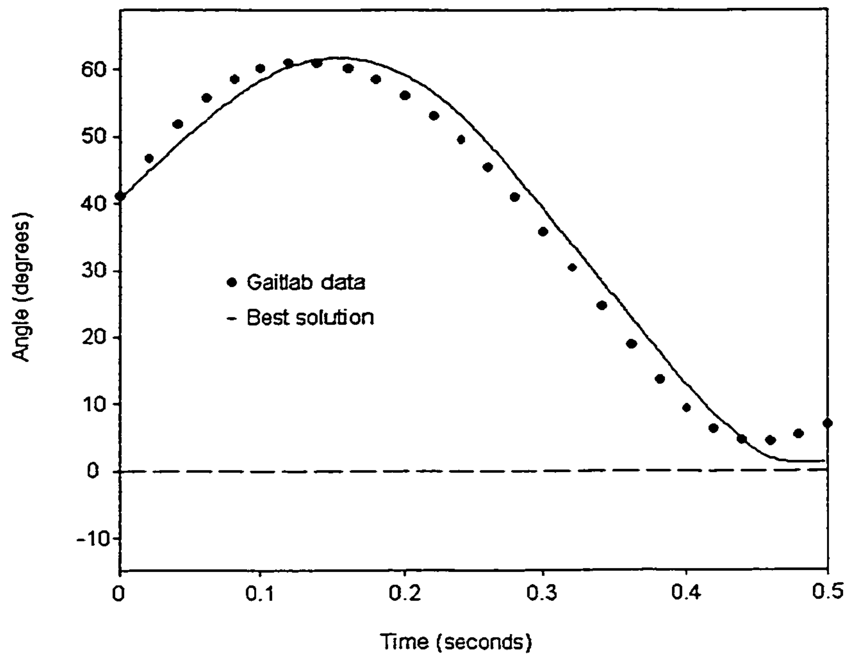

Figure 5.6: Best solution found for the three variable search test having parameters K_{assist} = -0.6471 Nm/radian, C_{assist} = 1.2549 Nms/radian and β_{damper} = 0.825 degrees for an RMS tracking error of 18.56 degrees. Shown as well is Gaitlab data from a real human.

	V_1	V_2	V_3
K_{assist}	-0.6863	-0.6275	-0.5882
C_{assist}	1.3137	1.2353	1.1961
β_{damper}	1.295	0.705	0.120
e_{RMS}	18.57	18.61	18.69

Table 5.6: High quality variant solutions V_1, V_2 and V_3 found for the three variable search.

ity, there is muscle activity that affects the motion of the leg during swing. Considering EMG data traces presented in Gaitlab, it can be seen that a hip flexor muscle group, the illiopsoas group, is active in the early phase of swing. As hip flexion generally results in knee flexion, it is natural to think that the addition of a hip flexor element in our model could improve the early phase of the knee flexion curve.

In a similar fashion, a modified knee locking mechanism might result in the model and reference graphs looking more similar in the region before heelstrike. If we include into our search space some of the variables describing the knee locking mechanism (one of the K_{locker}, C_{locker} or β_{locker} variables, or a combination of the same) then we could let the GA search for the parameters of the knee locker that result in the most realistic looking gait.

5.5 Model modifications

Based on the observations from the general trends of our reference and best-found knee flexion curves, we will alter our model by including a simple hip flexor element. This element is intended to mimic the role of the illiopsoas muscle group in early swing phase. Descriptions of sophisticated mathematical muscle models that take into account nervous signal activation dynamics, muscle connection locations and tissue properties can be found in the literature (Yamaguchi and Zajac, 1990). For the sake of concentrating on the problem as a design method, we will depict our hip flexor model simply as an applied moment to the swing thigh, L_2, for the duration $t \leq 0.1s$. The 'on time' was chosen based on the approximate activity range shown in Gaitlab for the illiopsoas muscle group.

Holding the activation time constant, we will let the GA search for the optimal magnitude of the moment that results in the the most normal looking swing. This hip flexor moment, M_{hip}, will be of constant magnitude throughout the activation range and be in the range of $1Nm \leq M_{hip} \leq 6Nm$. The lower bound was chosen as it barely affects the motion of the model while the upper bound visibly affects the motion of the model. Upon finding an optimal value of hip torque, one could try to correlate this back to reality using data from Gaitlab.

In order to get a more realistic knee flexion curve in late swing, we will consider the knee locker mechanism activation angle, β_{locker}, as an additional variable in our GA search. The knee locker spring and damper constants, K_{locker} and C_{locker}, will be held constant while the activation angle will be allowed to vary through the range $1 \leq \beta_{locker} \leq 11$.

5.5.1 Results

The addition of two variables increases the size of our search landscape from $2^{3\times8} = 2^{24}$ possible solutions to $2^{5\times8} = 2^{40}$ possible solutions. The population size required for successful GA increases with increasing solution space size. In a similar fashion to that of Section 4.3.1, the population size for the five variable problem was increased incrementally and the problem was run repeatedly until a population size was found that would consistently converge on high quality solutions. It was found that a population size of 240 individuals was sufficient for consistent convergence.

The knee flexion graph from the best design found in the five variable search is shown in Figure 5.7. The best solution found had parameter values of K_{assist} = -1.6667 Nm/radian, C_{assist} = 1.4902 Nms/radian, β_{damper} = 11.14 degrees, M_{hip} = 5.8627 Nm and β_{locker} = 6.96 degrees for an RMS tracking error of 13.42 degrees. Comparing this solution to that found in our three variable experiment, we have decreased our tracking error score by approximately 28%, a significant amount. Visual comparison of the knee flexion curves shown in Figures 5.6 and 5.7 reveals that the five variable solution yields a more natural looking knee flexion curve, especially in the later stages of the swing phase. Hence if the amputee can maintain a constant hip torque of about 5.86 Nm for the first 0.1 seconds of swing phase, a limb having these parameters would yield a near-natural looking swing. This seems reasonable, as Winter (1990) reports peak hip torque values in excess

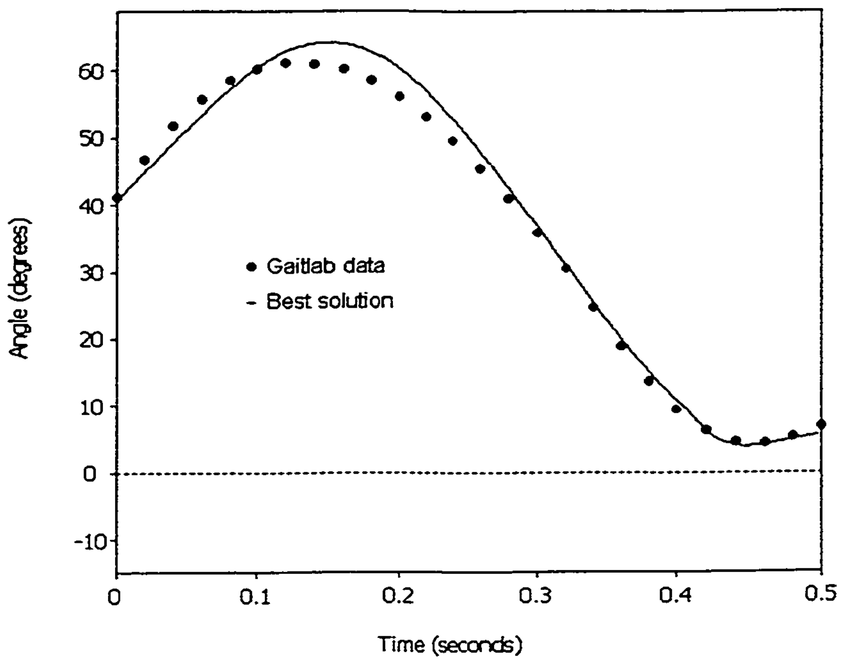

Figure 5.7: Best solution found for the five variable search test. Solid line is the best solution found having parameters K_{assist} = -1.6667 Nm/radian, C_{assist} = 1.4902 Nms/radian, β_{damper} = 11.14 degrees, M_{hip} = 5.8627 Nm and $\beta locker$ = 6.96 degrees for an RMS tracking error of 13.42 degrees. Filled circles represent real data taken from Gaitlab.

of $20Nm$ for a patient who has undergone a total hip replacement.

5.6 Conclusions

This chapter considered using the GA as a tool in a complex mechanism design task, particularly that of designing a purely mechanical above the knee prosthetic limb knee joint that exhibits natural looking motion in the swing phase of gait. The design aspect consisted of finding the constants for a rotary spring and damper attached to the swing knee joints and the active range of the damper element that resulted in a knee flexion curve similar to that of natural gait as observed in a real healthy male subject.

It was found that in order to realize 'convergence' with the GA, thereby indicating that a successful search has been performed, at least two items need careful consideration:

- An appropriate fitness function is required. By 'appropriate' it is implied that the quantities to be minimized/maximized are properly represented as mathematical equations.

- A population size is used that is large enough to consistently find the optimal solution set in the design space. In order to gauge consistency it is very useful to run the GA several times using different population sizes for a known problem.

Chapter 6

Conclusions and Recommendations

This thesis considers using the Genetic Algorithm as a tool for mechanical design tasks. Using appropriate fitness functions, the design tasks were formulated as optimization problems which the GA was used to solve.

The behaviour of the GA was first analyzed by considering different operating parameters. Varying the population size, crossover probability and mutation probability while monitoring the algorithm performance indicated suitable values for these input parameters to use in subsequent tests.

Using these parameter value guidelines, a 'real' example problem was considered that involved dimensional synthesis of a four bar mechanism for path generation tasks. Experimenting with different fitness function forms indicated the importance of formulating a fitness function that is indicative of the design task. It was shown that results can differ significantly depending on the fitness function formulation, often producing results which satisfy the objective function but don't satisfy the design task for other reasons.

The technique was then applied to another design task, that of designing a purely mechanical prosthetic knee joint. The objective function formulation considerations of the previous experiment were noted and optimal designs were found. In order to realize better results, the experiments were performed again with more of the model parameters considered to be design variables.

This increased the size of the solution space to be searched by the GA, and, with appropriate GA parameters, a better design was found in the larger design space. This was a successful example of starting with a small problem and adding complexity bit by bit to eventually solve a complex problem.

The GA is a suitable tool for mechanism design tasks, typically yielding optimal solutions when appropriate input parameters are used for a given design space. While reporting optimal solutions, the algorithm provides excellent insight whether or not the chosen objective function describes the design task. Objective function definitions not representing the design task are rapidly exposed, typically yielding mechanisms with obvious shortcomings.

In order to realize success using the GA, it was found that a population with enough individuals was required to effectively sample the solution space. The GA was relatively insensitive to local maxima, typically similar solutions were found for the design tasks when the GA was run several times on the task. The results of the optimization are only as good as the representation of the objective function for the design task. Several runs of the GA are required to increase the probability of finding the global optimum solution. A single run of the GA is not a wise approach.

The GA is computationally intensive, evaluating many solutions in the design space in a single run. As a population matures, many of the individuals have the same genetic makeup. It doesn't make sense to re-evaluate previously found individuals, so a means to store the results is a necessity if the objective function evaluations are expensive. Using a hash table structure is an efficient means of doing this and should be part of any GA.

Given an arbitrary design task, this GA design approach can be nearly a 'black box': an objective function, a set of design variables and the inputs for GA are all that is required. Assuming that a designer is familiar with their objective and has determined their design variables, the only thing left to specify is the input parameter set for the GA. No exact rules have been defined in this thesis for determining what the GA inputs should be. A methodology similar to that used in Section 4.4.1, however, can be utilized to determine approximate input parameters.

As determining the input parameters for a given design space size and complexity is the most difficult aspect of using the GA, future work might include trying to come up with general formulas for determining the population size and probabilities required. Alternatively a means of adjusting the GA parameters throughout a simulation to react to population characteristics would be helpful. A GA that could adjust its own parameters to maximize search performance would be a big step towards having a 'black box' design method. This would enable designers considering large, complex solution spaces to use the GA as a design tool, avoiding much of the initial tuning runs required for a particular problem.

Appendix A

Finding four bar linkage position using the Newton Raphson root finding technique

A method is needed to find the mechanism configuration of a four bar linkage as its crank arm is driven through a complete revolution. The planar four bar linkage, as shown in Figure A.1, is a 1 Degree of Freedom (1DOF) mechanical system. Only 1 independent generalized coordinate is needed to uniquely specify the system configuration at any point. If this coordinate (the crank angle, ϕ, for this mechanism) is specified and it corresponds to a possible physical configuration of the mechanism, a mathematically determinate system should describe the physical system.

Having specified the value of the single independent coordinate, the orientation of the system should be fully defined. In order to find the coupler and follower link angles (β and θ), the kinematic constraint equations for the system must be written in terms of the generalized coordinate. In the general case this results in a system of n constraint equations written in terms of the n unknowns. The kinematic constraint equations for this system can be written as the x- and y- components of the loop closure equation for the four bar:

$$u = u(\beta, \theta) = L_1 cos(\phi) + L_2 cos(\beta) + L_3 cos(\theta) - Base_x = 0, \qquad \text{(A.1)}$$

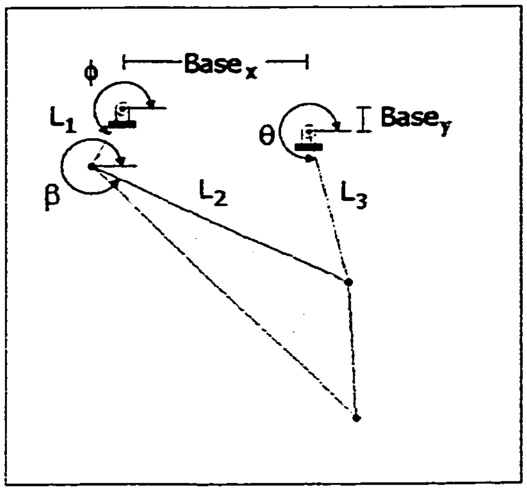

Figure A.1: Planar four bar linkage. Dimensions given as $L_1, L_2, L_3, Base_x$ and $Base_y$. Orientation specified by ϕ - β and θ are computed.

and

$$v = v(\beta,\theta) = L_1 sin(\phi) + L_2 sin(\beta) + L_3 sin(\theta) - Base_y = 0. \tag{A.2}$$

Specifying the crank angle ϕ yields a system of two equations and two unknowns, non-linear in terms of the unknowns β and θ. If a physical solution exists at a given crank angle, then this system should be mathematically determinate.

Due to the nonlinearity in the system, an analytical solution for the two unknowns may not be found. For this reason a numerical solution technique is employed to solve the system of equations. The Newton Raphson numerical root finding technique will be used to refine initial guesses for the β and θ values at each crank angle.

The Newton Raphson technique can be found in most introductory numerical analysis textbooks (Chapra and Canale (1988), Press *et al.* (1988)) but is included here for completeness. The 1st order Taylor series approximation of a function of one variable, x, at a point is:

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i). \tag{A.3}$$

If the function intersects the x-axis at the next step, ie: $f(x_{i+1}) = 0$, this equation can be rewritten as the equality:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \tag{A.4}$$

Equation A.4 provides a means to refine a root estimate: given a guess for a root value (x_i) and the function and first derivative values at this point, a better estimate is predicted with this information.

This process of improving on the root estimates is shown graphically in Figure A.1. Root values should become progressively closer to the true root value. The accuracy of a root value is determined by evaluating the function at that point. If the following conditions are met:

- The new root estimate is changing slowly ($x_{i+1} - x_i < \epsilon_x$, where ϵ_x is an estimate difference tolerance).
- The new function value is less than some error tolerance ($|f(x_{i+1})| < \epsilon_f$, where ϵ_f is the amount the function value may deviate from zero.

Then x_{i+1} is taken to be a root of the function.

The loop closure equations (Equ. A.1 and A.2), are a system of two unknowns and two equations. Following a similar process the Newton-Raphson method is derived for a system of two unknowns (β and θ) and two equations ($u(\beta, \theta)$ and $v(\beta, \theta)$) to yield:

$$\beta_{i+1} = \beta_i - \frac{u_i \frac{\partial v_i}{\partial \theta} - v_i \frac{\partial u_i}{\partial \theta}}{\frac{\partial u_i}{\partial \beta}\frac{\partial v_i}{\partial \theta} - \frac{\partial u_i}{\partial \theta}\frac{\partial v_i}{\partial \beta}} \tag{A.5}$$

and

$$\theta_{i+1} = \theta_i + \frac{u_i \frac{\partial v_i}{\partial \beta} - v_i \frac{\partial u_i}{\partial \beta}}{\frac{\partial u_i}{\partial \beta}\frac{\partial v_i}{\partial \theta} - \frac{\partial u_i}{\partial \theta}\frac{\partial v_i}{\partial \beta}}. \tag{A.6}$$

Rewriting the loop closure equations in the form of Equations A.5 and A.6 therefore provides an estimate for new β and θ values given an initial guess for each. Initial guesses for the β and θ angles at each crank angle can

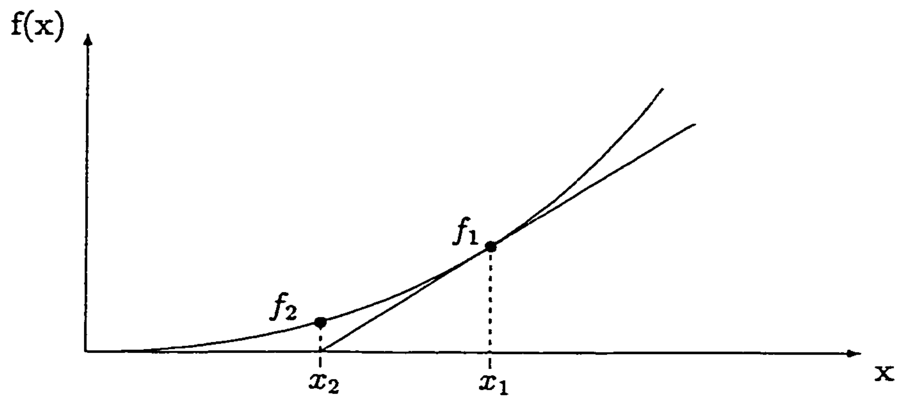

Figure A.2: Graphical representation of Newton Raphson root finding technique.

be conveniently set to the angles at the previous crank angle: assuming that the mechanism configuration changes smoothly and the crank angle step size is small enough, in practice this is a valid method for finding the new crank angles.

Appendix B

Finding the circle defined by three points in a plane

Three points in a plane specify a unique circle, save for the special case where the three points lie in a straight line. The center of this circle is located at the intersection point of the perpendicular bisectors of two lines that join successive specified points. This is shown graphically in Figure B.1.

Specified points $P1$, $P2$ and $P3$ are on the circumference of the circle. The successive joining lines l_1 and l_2 connect points $P1$ and $P2$ and $P2$ and $P3$, respectively. The midpoints, $Mid(1)$ and $Mid(2)$, of the two joining lines are found by the equations:

$$Mid(i)_x = \frac{P(i)_x + P(i+1)_x}{2} \tag{B.1}$$

$$Mid(i)_y = \frac{P(i)_y + P(i+1)_y}{2}. \tag{B.2}$$

where $P(i)_x$ is the x-component of the ith point and i is a point index.

Lines l_1 and l_2 have slopes m_1 and m_2, respectively, found by the equation:

$$m_i = \frac{P(i+1)_y - P(i)_y}{P(i+1)_x - P(i)_x} \tag{B.3}$$

The slope of a perpendicular bisector line is the negative reciprocal of the reference line l_i. Knowing the slope of the line and the location of a point on

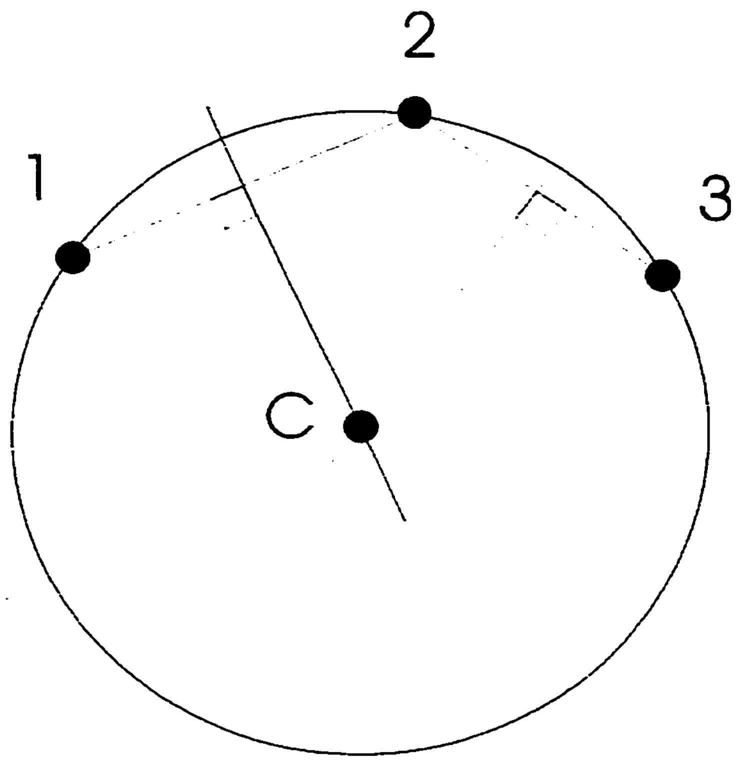

Figure B.1: Finding the unique circle that passes through three points in a plane.

the line, the equation of each perpendicular bisector line is found as:

$$y - Mid(i)_y = -\frac{1}{m_i}(x - Mid(i)_x). \tag{B.4}$$

Finding the equations of l_{p1} and l_{p2} in this fashion allows one to solve for the location of the intersecting x by equating the common y-values. This results in an equation for x of the form:

$$x = \frac{y_3 - y_1 - A(x_1 + x_2) + B(x_2 + x_3)}{2B - 2A} \tag{B.5}$$

which can then be substituted into Equation B.4 to find the y coordinate of the circle center.

Using this technique, a circle is fitted to three points along an arbitrary curve. If an approximation is required for the smallest distance that another point lies off of the curve, an efficient approximation to this distance can then be determined by first calculating the distance from the outlying point to the circle center point, then finding the difference knowing the constructed circle's radius.

Appendix C

Equations of motion for a three link pendulum

The equations of motion for a three link planar pendulum system having three moving bodies and three revolute joints was found using Newton's 2nd Law, ($\vec{F} = m\vec{a}$). A three link pendulum is shown in a deformed position in Figure C.1. In terms of the pendulum's three coordinates, ϕ_1, ϕ_2 and ϕ_3, the equations of motion are written as:

$$[I]\{\ddot{\phi}\} + [M]\{\dot{\phi}^2\} + \{G\} = \{0\} \tag{C.1}$$

where:

$$[I] = \begin{bmatrix} I_1 + L_1^2(\frac{m_1}{4} + m_2 + m_3) & L_1L_2(\frac{m_2}{2} + m_3)cos(\phi_1 - \phi_2) & L_1L_3(\frac{m_3}{2})cos(\phi_3 - \phi_1) \\ L_1L_2(\frac{m_2}{2} + m_3)cos(\phi_1 - \phi_2) & I_2 + L_2^2(\frac{m_2}{4} + m_3) & L_2L_3(\frac{m_3}{2})cos(\phi_3 - \phi_2) \\ L_1L_3(\frac{m_3}{2})cos(\phi_3 - \phi_1) & L_2L_3(\frac{m_3}{2})cos(\phi_3 - \phi_2) & I_3 + \frac{m_3L_3^2}{4} \end{bmatrix} \tag{C.2}$$

and:

$$[M] = \begin{bmatrix} 0 & L_1L_2(\frac{m_2}{2} + m_3)sin(\phi_1 - \phi_2) & L_1L_3(\frac{m_3}{2})sin(\phi_1 - \phi_3) \\ -L_1L_2(\frac{m_2}{2} + m_3)sin(\phi_1 - \phi_2) & 0 & -L_2L_3(\frac{m_3}{2})sin(\phi_3 - \phi_2) \\ -L_1L_3(\frac{m_3}{2})sin(\phi_1 - \phi_3) & L_2L_3(\frac{m_3}{2})sin(\phi_3 - \phi_2) & 0 \end{bmatrix} \tag{C.3}$$

and:

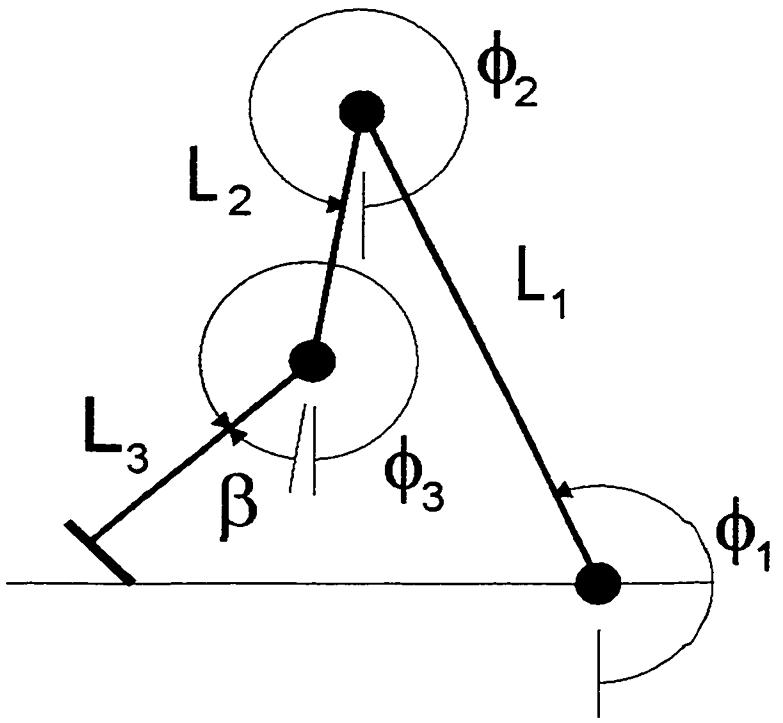

Figure C.1: Three link pendulum. Angular coordinates are measured from Cartesian reference frame.

$$[G] = \begin{bmatrix} gL_1 sin\phi_1(\frac{m_1}{2} + m_2 + m_3) \\ gL_2 sin\phi_2(\frac{m_2}{2} + m_3) \\ gL_3 sin\phi_3(\frac{m_3}{2}) \\ \end{bmatrix} \tag{C.4}$$

Bibliography

Chapra, S. C. and R. P. Canale (1988). *Numerical Methods for Engineers* (Second ed.). McGraw-Hill Publishing Company.

Davy, D. T. and M. L. Audu (1987). A dynamic optimization technique for predicting muscle forces in the swing phase of gait. *Journal of Biomechanics 20*, 187–201.

Goldberg, D. E. (1989a). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company.

Goldberg, D. E. (1989b). Sizing populations for serial and parallel genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 70–79.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics SMC-16*(1), 122–128.

Kim, C. H. (1995). *Coupler Point Path Synthesis Using a Hybrid Genetic Algorithm.* Ph. D. thesis, Department of Mechanical Engineering, University of Utah.

Mallik, A. K., A. Ghosh, and G. Dittrich (1994). *Kinematic Analysis and Synthesis of Mechanisms.* CRC Press.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms.* The MIT Press.

Mochon, S. and T. A. McMahon (1980). Ballistic walking. *Journal of*

Biomechanics 13, 49–57.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (1988). *Numerical Recipes in C* (Second ed.). Cambridge University Press.

Roston, G. P. and R. H. Sturges (1996). Genetic algorithm synthesis of four-bar mechanisms. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10*, 371–390.

Sandor, G. N. and A. G. Erdman (1984). *Advanced Mechanism Design: Analysis and Synthesis*, Volume 2. Prentice-Hall Inc.

Segla, S., C. M. Kalker-Kalkman, and A. Schwab (1998). Statical balancing of a robot mechanism with the aid of a genetic algorithm. *Mechanism and Machine Theory 33*(1), 163–174.

Vaughn, C. L., B. L. Davis, and J. C. O'Connor (1992). *Gait Analysis Laboratory*. Human Kinetics Publishers.

Wampler, C. W., A. P. Morgan, and A. J. Sommese (1992). Complete solution of the nine-point path synthesis problem for four-bar linkages. *Journal of Mechanical Design SMC-16*(114), 153–159.

Winter, D. A. (1990). *Biomechanics and Motor Control of Human Movement, Second Edition*. John Wiley and Sons, Inc.

Yamaguchi, G. T. and F. E. Zajac (1990). Restoring unassisted natural gait to paraplegics via function neuromuscular stimulation: a computer simulation study. *IEEE Transactions on Biomedical Engineering 37*, 886–902.